**\*\*\*\*\*\*\*\*\*\* EXPERIMENT: 05 \*\*\*\*\*\*\*\*\*\***

**Aim:** To write PL/SQL blocks (with exception handling) using 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)

**Author : Mehul Y Khandhadiya**

**Roll no : 55**

**Date : 25-Sep-2020**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 01 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write SQL code to create and execute an anonymous PL/SQL block that will insert 5 tuples into EXAM. Ensure to commit the populated records. Test theinsertion in EXAM by displaying its contents.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CREATE TABLE EXAM (**

**UROLL NUMBER(4) NOT NULL,**

**COURSE VARCHAR2(25) NOT NULL,**

**EXAMDT DATE NOT NULL,**

**CONSTRAINTS EXAM\_PK\_UROLL PRIMARY KEY (UROLL),**

**CONSTRAINTS EXAM\_CK\_UROLL CHECK (UROLL>=1001 AND UROLL<=1099)**

**);**

Table created.

**INSERT INTO EXAM VALUES(1031, 'DBMS', '24-SEP-2020');**

**1 row created.**

**SELECT \* FROM EXAM;**

UROLL COURSE EXAMDT

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

1031 DBMS 24-SEP-20

**SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, TABLE\_NAME**

**FROM USER\_CONSTRAINTS**

**WHERE TABLE\_NAME IN ('EXAM');**

CONSTRAINT\_NAME C TABLE\_NAME

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

SYS\_C0012383 C EXAM

SYS\_C0012384 C EXAM

SYS\_C0012385 C EXAM

EXAM\_CK\_UROLL C EXAM

EXAM\_PK\_UROLL P EXAM

**BEGIN**

**INSERT INTO EXAM VALUES(1032,'DSA','24-SEP-2020');**

**INSERT INTO EXAM VALUES(1033,'CN','25-SEP-2020');**

**INSERT INTO EXAM VALUES(1034,'DAA','26-SEP-2020');**

**INSERT INTO EXAM VALUES(1035,'DP','29-SEP-2020');**

**INSERT INTO EXAM VALUES(1036,'CG','21-SEP-2020');**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('5 TUPLES INSERTED SUCCESSFULLY');**

**END;**

**/**

5 TUPLES INSERTED SUCCESSFULLY

PL/SQL procedure successfully completed.

**SELECT \* FROM EXAM;**

UROLL COURSE EXAMDT

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

1031 DBMS 24-SEP-20

1032 DSA 24-SEP-20

1033 CN 25-SEP-20

1034 DAA 26-SEP-20

1035 DP 29-SEP-20

1036 CG 21-SEP-20

6 rows selected.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 02 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write SQL code to create and execute an anonymous PL/SQL block that will use %TYPE variables to populate the EMPP table with corresponding tuples in EMPLOYEE table.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**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 CONSTRAINTS EMPP\_PK\_EID PRIMARY KEY(EID);**

Table altered.

**DECLARE**

**EID EMPLOYEE.ENO%TYPE;**

**ENAME EMPP.ENAME%TYPE;**

**HIREDT EMPLOYEE.HIREDATE%TYPE;**

**DESIG EMPLOYEE.DESIGNATION%TYPE;**

**SAL EMPLOYEE.SALARY%TYPE;**

**KNT NUMBER(4) ;**

**BEGIN**

**FOR KNT IN 7101 .. 7117 LOOP**

**SELECT ENO, FNAME||' '||LNAME, HIREDATE, DESIGNATION, SALARY**

**INTO EID, ENAME, HIREDT, DESIG, SAL FROM EMPLOYEE**

**WHERE ENO=KNT;**

**INSERT INTO EMPP**

**VALUES (EID,ENAME,HIREDT,DESIG,SAL);**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('RECORDS INSERTED SUCCESSFULLY...');**

**END;**

**/**

RECORDS INSERTED SUCCESSFULLY...

PL/SQL procedure successfully completed**.**

**SELECT \* FROM EMPP;**

EID ENAME HIREDATE DESIGNATION SALARY

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

7101 Eugene Sabatini 10-OCT-06 Professor 150000

7102 Samantha Jones 08-NOV-06 Professor 146500

7103 Alexander Lloyd 01-FEB-07 Professor 148000

7104 Simon Downing 01-SEP-07 Professor 138400

7105 Christina Mulboro 15-JUL-08 Asso. Professor 127400

7106 Dolly Silverline 17-AUG-08 Asso. Professor 127400

7107 Christov Plutnik 01-SEP-08 Asso. Professor 127400

7108 Ellena Sanchez 12-NOV-09 Asso. Professor 119700

7109 Martina Jacobson 15-NOV-09 Asst. Professor 91000

7110 William Smithfield 23-JUN-10 Asst. Professor 86400

7111 Albert Greenfield 12-JUL-16 Research Asst. 48200

7112 James Washington 22-AUG-17 Research Asst. 44600

7113 Julia Martin 01-DEC-18 Teaching Asst. 35600

7114 Larry Gomes 18-MAY-19 Teaching Asst. 32850

7115 Svetlana Sanders 15-JAN-20 Teaching Asst. 30000

7116 Lovelyn Brendon 17-JUL-20 Teaching Asst. 30000

7117 Hector Hercules 01-AUG-20 Teaching Asst. 32200

17 rows selected.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 03 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

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

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CREATE TABLE MENTEE**

**AS (SELECT SID AS STAFF\_NUMBER, NAME AS STAFF\_NAME, FNAME||' '||LNAME AS STUDENT\_NAME,**

**ROLL AS ROLL\_NO, REG\_DT AS REGISTRATION\_DATE**

**FROM STUDENT, STAFF**

**WHERE 1=2);**

Table created.

**ALTER TABLE MENTEE**

**ADD CONSTRAINT MENTEE\_PK\_SNO\_RNO PRIMARY KEY (STAFF\_NUMBER, ROLL\_NO);**

Table altered.

**SELECT \* FROM MENTEE;**

no rows selected

**DESC MENTEE;**

Name Null? Type

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

STAFF\_NUMBER NOT NULL NUMBER(3)

STAFF\_NAME NOT NULL VARCHAR2(25)

STUDENT\_NAME VARCHAR2(31)

ROLL\_NO NOT NULL NUMBER(3)

REGISTRATION\_DATE NOT NULL DATE

**SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, TABLE\_NAME**

**FROM USER\_CONSTRAINTS**

**WHERE TABLE\_NAME IN ('MENTEE');**

CONSTRAINT\_NAME C TABLE\_NAME

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

SYS\_C0012462 C MENTEE

SYS\_C0012463 C MENTEE

SYS\_C0012464 C MENTEE

MENTEE\_PK\_SNO\_RNO P MENTEE

4 rows selected.

**DECLARE**

**R\_MENTEE MENTEE%ROWTYPE;**

**KNT NUMBER(2);**

**BEGIN**

**KNT:=1;**

**WHILE KNT<=87 LOOP**

**SELECT SID,NAME,FNAME||' '||LNAME,ROLL,REG\_DT**

**INTO R\_MENTEE FROM STUDENT,STAFF**

**WHERE ROLL=KNT AND SID=ADVISOR;**

**INSERT INTO MENTEE VALUES R\_MENTEE;**

**KNT:=KNT+1;**

**IF KNT=20 THEN**

**KNT:=30;**

**ELSIF KNT=49 THEN**

**KNT:=51;**

**ELSIF KNT=69 THEN**

**KNT:=71;**

**END IF;**

**END LOOP;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('INSERTION DONE SUCCESSFULLY...');**

**END;**

INSERTION DONE SUCCESSFULLY...

PL/SQL procedure successfully completed.

**SELECT \* FROM MENTEE;**

STAFF\_NO STAFF\_NAME STUDENT\_NAME ROLL\_NO REG\_DT

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

101 Kamalkant Marathe Afra Sayed 1 20-JUL-18

104 Aasawari Deodhar Akansha Wasalu 2 20-JUL-18

108 Harmeet Khullar Anjali Rajendran 3 19-JUL-18

109 Jasmine Arora Aradhita Menghal 4 07-JUL-18

102 Adishesh Vidyarthi Ketki Fadnavis 5 14-JUL-18

110 Sanjeev Bamireddy Lalita Sharma 6 10-JUL-18

103 Manishi Singh Muskan Gupta 7 19-JUL-18

106 DEO NARAYAN MISHRA Prateeksha Devikar 8 13-JUL-18

105 Geetika Goenka Priyal Taori 9 19-JUL-18

107 Vallabh Pai Rashi Chouksey 10 08-AUG-18

101 Kamalkant Marathe Ritul Deshmukh 11 18-JUL-18

104 Aasawari Deodhar Sakshi Nema 12 07-JUL-18

108 Harmeet Khullar Shreya Agnihotri 13 07-JUL-18

109 Jasmine Arora Shrishti Shukla 14 19-JUL-18

102 Adishesh Vidyarthi Simran Baheti 15 20-JUL-18

110 Sanjeev Bamireddy Urvi Negi 16 19-JUL-18

106 DEO NARAYAN MISHRA Deepali Pathe 17 10-AUG-19

103 Manishi Singh Prachi Bhanuse 18 11-AUG-19

107 Vallabh Pai Siddhi Tripathi 19 31-AUG-19

109 Jasmine Arora Renuka Soni 30 25-JUL-16

101 Kamalkant Marathe Aayush Muley 31 19-JUL-18

104 Aasawari Deodhar Abhishek Chohan 32 07-JUL-18

108 Harmeet Khullar Adesh Kotgirwar 33 20-JUL-18

109 Jasmine Arora Adhney Nawghare 34 08-AUG-18

102 Adishesh Vidyarthi Akshat Chandak 35 20-JUL-18

110 Sanjeev Bamireddy Amey Chole 36 08-AUG-18

103 Manishi Singh Amit Ray 37 20-JUL-18

106 DEO NARAYAN MISHRA Aryan Pandharipande 38 07-JUL-18

105 Geetika Goenka Atharva Uplanchiwar 39 07-JUL-18

107 Vallabh Pai Atharva Paliwal 40 20-JUL-18

101 Kamalkant Marathe Ayush Gupta 41 12-JUL-18

104 Aasawari Deodhar Chaitanya Kapre 42 25-JUL-18

108 Harmeet Khullar Dev Paliwal 43 21-JUL-18

109 Jasmine Arora Gaurav Shukla 44 17-JUL-18

110 Sanjeev Bamireddy Gursewak Virdi 45 07-JUL-18

102 Adishesh Vidyarthi Saurabh Khandagale 46 10-AUG-19

106 DEO NARAYAN MISHRA Ganesh Thakur 47 22-AUG-19

103 Manishi Singh Manishkumar Pardhi 48 23-AUG-19

105 Geetika Goenka Harsh Karwa 51 11-JUL-18

107 Vallabh Pai Jayesh Kapse 52 08-AUG-18

109 Jasmine Arora Keshubh Sharma 53 20-JUL-18

108 Harmeet Khullar Kunal Thorane 54 08-AUG-18

104 Aasawari Deodhar Mehul Khandhadiya 55 19-JUL-18

101 Kamalkant Marathe Nikhil Tiwari 56 04-JUL-18

102 Adishesh Vidyarthi Paritosh Dandekar 57 14-JUL-18

110 Sanjeev Bamireddy Pavankumar Gupta 58 03-JUL-18

103 Manishi Singh Rahul Agrawal 59 16-JUL-18

106 DEO NARAYAN MISHRA Rajat Chandak 60 20-JUL-18

107 Vallabh Pai Ram Agrawal 61 19-JUL-18

105 Geetika Goenka Raunak Khandelwal 62 19-JUL-18

104 Aasawari Deodhar Rishikesh Kale 63 07-JUL-18

108 Harmeet Khullar Ritik Parashar 64 19-JUL-18

101 Kamalkant Marathe Rohit Chandani 65 08-AUG-18

107 Vallabh Pai Ayush Singh 66 27-JUL-17

103 Manishi Singh Anujesh Soni 67 25-JUL-17

110 Sanjeev Bamireddy Love Sharnagat 68 25-JUL-17

110 Sanjeev Bamireddy Rushil Parikh 71 07-JUL-18

102 Adishesh Vidyarthi Sankalp Pandey 72 07-JUL-18

103 Manishi Singh Saurabh Sushir 73 07-JUL-18

106 DEO NARAYAN MISHRA Shardul Nimbalkar 74 28-JUL-17

105 Geetika Goenka Shashank Tapas 75 07-JUL-18

107 Vallabh Pai Shivam Bagadia 76 20-JUL-18

105 Geetika Goenka Shreyas Nemani 77 20-JUL-18

109 Jasmine Arora Shubham Jha 78 12-JUL-18

108 Harmeet Khullar Yaman Kushwah 79 17-JUL-18

104 Aasawari Deodhar Yash Bhageriya 80 19-JUL-18

102 Adishesh Vidyarthi Yash Daware 81 20-JUL-18

110 Sanjeev Bamireddy Yash Roy 82 07-JUL-18

106 DEO NARAYAN MISHRA Yash Dhamecha 83 21-JUL-18

103 Manishi Singh Yash Jain 84 03-JUL-18

105 Geetika Goenka Yogesh Siral 85 21-JUL-18

107 Vallabh Pai Shapath Pandey 86 27-JUL-17

108 Harmeet Khullar Mayank Rangari 87 25-JUL-16

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 04 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write SQL code to create 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(Q), while including proper headers in the result.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('STAFF\_NO',8,' ')||' '||**

**LPAD('STAFF\_NAME',20,' ')||' '||LPAD('STUDENT\_NAME',20,' ')||' '||**

**LPAD('ROLL\_NO',7,' ')||' '||LPAD('REG\_DT',10,' '));**

**DBMS\_OUTPUT.PUT\_LINE(LPAD(' ',8,'-')||' '||LPAD(' ',20,'-')||' '||**

**LPAD(' ',20,'-')||' '||LPAD(' ',7,'-')||' '||LPAD(' ',10,'-'));**

**FOR t IN (SELECT \* FROM MENTEE) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(LPAD(t.STAFF\_NUMBER,8,' ')||' '||**

**LPAD(t.STAFF\_NAME,25,' ')||' '||LPAD(t.STUDENT\_NAME,20,' ')**

**||' '||LPAD(t.ROLL\_NO,7,' ')||' '||LPAD(t.REGISTRATION\_DATE,10,' '));**

**END LOOP;**

**END;/**

STAFF\_NO STAFF\_NAME STUDENT\_NAME ROLL\_NO REG\_DT

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

101 Kamalkant Marathe Afra Sayed 1 20-JUL-18

104 Aasawari Deodhar Akansha Wasalu 2 20-JUL-18

108 Harmeet Khullar Anjali Rajendran 3 19-JUL-18

109 Jasmine Arora Aradhita Menghal 4 07-JUL-18

102 Adishesh Vidyarthi Ketki Fadnavis 5 14-JUL-18

110 Sanjeev Bamireddy Lalita Sharma 6 10-JUL-18

103 Manishi Singh Muskan Gupta 7 19-JUL-18

106 DEO NARAYAN MISHRA Prateeksha Devikar 8 13-JUL-18

105 Geetika Goenka Priyal Taori 9 19-JUL-18

107 Vallabh Pai Rashi Chouksey 10 08-AUG-18

101 Kamalkant Marathe Ritul Deshmukh 11 18-JUL-18

104 Aasawari Deodhar Sakshi Nema 12 07-JUL-18

108 Harmeet Khullar Shreya Agnihotri 13 07-JUL-18

109 Jasmine Arora Shrishti Shukla 14 19-JUL-18

102 Adishesh Vidyarthi Simran Baheti 15 20-JUL-18

110 Sanjeev Bamireddy Urvi Negi 16 19-JUL-18

106 DEO NARAYAN MISHRA Deepali Pathe 17 10-AUG-19

103 Manishi Singh Prachi Bhanuse 18 11-AUG-19

107 Vallabh Pai Siddhi Tripathi 19 31-AUG-19

109 Jasmine Arora Renuka Soni 30 25-JUL-16

101 Kamalkant Marathe Aayush Muley 31 19-JUL-18

104 Aasawari Deodhar Abhishek Chohan 32 07-JUL-18

108 Harmeet Khullar Adesh Kotgirwar 33 20-JUL-18

109 Jasmine Arora Adhney Nawghare 34 08-AUG-18

102 Adishesh Vidyarthi Akshat Chandak 35 20-JUL-18

110 Sanjeev Bamireddy Amey Chole 36 08-AUG-18

103 Manishi Singh Amit Ray 37 20-JUL-18

106 DEO NARAYAN MISHRA Aryan Pandharipande 38 07-JUL-18

105 Geetika Goenka Atharva Uplanchiwar 39 07-JUL-18

107 Vallabh Pai Atharva Paliwal 40 20-JUL-18

101 Kamalkant Marathe Ayush Gupta 41 12-JUL-18

104 Aasawari Deodhar Chaitanya Kapre 42 25-JUL-18

108 Harmeet Khullar Dev Paliwal 43 21-JUL-18

109 Jasmine Arora Gaurav Shukla 44 17-JUL-18

110 Sanjeev Bamireddy Gursewak Virdi 45 07-JUL-18

102 Adishesh Vidyarthi Saurabh Khandagale 46 10-AUG-19

106 DEO NARAYAN MISHRA Ganesh Thakur 47 22-AUG-19

103 Manishi Singh Manishkumar Pardhi 48 23-AUG-19

105 Geetika Goenka Harsh Karwa 51 11-JUL-18

107 Vallabh Pai Jayesh Kapse 52 08-AUG-18

109 Jasmine Arora Keshubh Sharma 53 20-JUL-18

108 Harmeet Khullar Kunal Thorane 54 08-AUG-18

104 Aasawari Deodhar Mehul Khandhadiya 55 19-JUL-18

101 Kamalkant Marathe Nikhil Tiwari 56 04-JUL-18

102 Adishesh Vidyarthi Paritosh Dandekar 57 14-JUL-18

110 Sanjeev Bamireddy Pavankumar Gupta 58 03-JUL-18

103 Manishi Singh Rahul Agrawal 59 16-JUL-18

106 DEO NARAYAN MISHRA Rajat Chandak 60 20-JUL-18

107 Vallabh Pai Ram Agrawal 61 19-JUL-18

105 Geetika Goenka Raunak Khandelwal 62 19-JUL-18

104 Aasawari Deodhar Rishikesh Kale 63 07-JUL-18

108 Harmeet Khullar Ritik Parashar 64 19-JUL-18

101 Kamalkant Marathe Rohit Chandani 65 08-AUG-18

107 Vallabh Pai Ayush Singh 66 27-JUL-17

103 Manishi Singh Anujesh Soni 67 25-JUL-17

110 Sanjeev Bamireddy Love Sharnagat 68 25-JUL-17

110 Sanjeev Bamireddy Rushil Parikh 71 07-JUL-18

102 Adishesh Vidyarthi Sankalp Pandey 72 07-JUL-18

103 Manishi Singh Saurabh Sushir 73 07-JUL-18

106 DEO NARAYAN MISHRA Shardul Nimbalkar 74 28-JUL-17

105 Geetika Goenka Shashank Tapas 75 07-JUL-18

107 Vallabh Pai Shivam Bagadia 76 20-JUL-18

105 Geetika Goenka Shreyas Nemani 77 20-JUL-18

109 Jasmine Arora Shubham Jha 78 12-JUL-18

108 Harmeet Khullar Yaman Kushwah 79 17-JUL-18

104 Aasawari Deodhar Yash Bhageriya 80 19-JUL-18

102 Adishesh Vidyarthi Yash Daware 81 20-JUL-18

110 Sanjeev Bamireddy Yash Roy 82 07-JUL-18

106 DEO NARAYAN MISHRA Yash Dhamecha 83 21-JUL-18

103 Manishi Singh Yash Jain 84 03-JUL-18

105 Geetika Goenka Yogesh Siral 85 21-JUL-18

107 Vallabh Pai Shapath Pandey 86 27-JUL-17

108 Harmeet Khullar Mayank Rangari 87 25-JUL-16

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 05 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write SQL code to create 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.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**DECLARE**

**TODAY VARCHAR2(7);**

**BEGIN**

**TODAY:=SYSDATE;**

**DBMS\_OUTPUT.PUT\_LINE('DATE IS: '||TO\_CHAR(TODAY));**

**EXCEPTION**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('Variable size is not sufficient');**

**END;**

**/**

Variable size is not sufficient

PL/SQL procedure successfully completed.

**DECLARE**

**TODAY VARCHAR2(10);**

**BEGIN**

**TODAY:=SYSDATE;**

**DBMS\_OUTPUT.PUT\_LINE('DATE IS: '||TO\_CHAR(TODAY));**

**EXCEPTION**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('Variable size is not sufficient');**

**END;**

**/**

DATE IS: 25-SEP-20

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 06 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write 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 year of service. Now, re-execute the block to extend longevity bonus to employees with 10 years of service.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**BEGIN**

**FOR t IN (SELECT \* FROM EMPLOYEE) LOOP**

**IF EXTRACT(YEAR FROM TO\_DATE(t.HIREDATE, 'DD-MON-RR')) <=2008 THEN**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NAMED'||' '||t.FNAME||' '||t.LNAME||' IS ELIGIBLE TO RECIEVE LONGVITY BONUS');**

**END IF;**

**END LOOP;**

**END;**

**/**

EMPLOYEE NAMED Samantha Jones IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Eugene Sabatini IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Alexander Lloyd IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Simon Downing IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Christov Plutnik IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Christina Mulboro IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Dolly Silverline IS ELIGIBLE TO RECIEVE LONGVITY BONUS

PL/SQL procedure successfully completed.

**LIST**

**BEGIN**

**FOR t IN (SELECT \* FROM EMPLOYEE) LOOP**

**IF EXTRACT(YEAR FROM TO\_DATE(t.HIREDATE, 'DD-MON-RR')) <=2008 THEN**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NAMED'||' '||t.FNAME||' '||t.LNAME||' IS ELIGIBLE TO RECIEVE LONGVITY BONUS');**

**END IF;**

**END LOOP;**

**\* END;**

**L3**

**3\* IF EXTRACT(YEAR FROM TO\_DATE(t.HIREDATE, 'DD-MON-RR')) <=2008 THEN**

**C/2008/2010;**

**3\* IF EXTRACT(YEAR FROM TO\_DATE(t.HIREDATE, 'DD-MON-RR')) <=2010 THEN**

**/**

EMPLOYEE NAMED Samantha Jones IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Eugene Sabatini IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Alexander Lloyd IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Simon Downing IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Christov Plutnik IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Christina Mulboro IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Dolly Silverline IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Ellena Sanchez IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED Martina Jacobson IS ELIGIBLE TO RECIEVE LONGVITY BONUS

EMPLOYEE NAMED William Smithfield IS ELIGIBLE TO RECIEVE LONGVITY BONUS

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 07 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write SQL cade 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 table is searched in reversed order.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**BEGIN**

**FOR t IN (SELECT \* FROM EMPLOYEE) LOOP**

**IF EXTRACT(MONTH FROM TO\_DATE(t.BIRTHDATE, 'DD-MON-RR')) = 08 THEN**

**DBMS\_OUTPUT.PUT\_LINE('THE FIRST AUGUST BORN EMPLOYEE IN THE EMPLOYEE SCHEMA IS'||' '||t.FNAME||' '||t.LNAME||' WITH BIRTHDATE'||' '||t.BIRTHDATE);**

**EXIT;**

**END IF;**

**END LOOP;**

**END;**

**/**

THE FIRST AUGUST BORN EMPLOYEE IN THE EMPLOYEE SCHEMA IS Larry Gomes WITH BIRTHDATE 28-AUG-96

PL/SQL procedure successfully completed.

**DECLARE**

**FNAME EMPLOYEE.FNAME%TYPE;**

**LNAME EMPLOYEE.LNAME%TYPE;**

**BD EMPLOYEE.BIRTHDATE%TYPE;**

**BEGIN**

**FOR t IN (SELECT \* FROM EMPLOYEE) LOOP**

**IF EXTRACT(MONTH FROM TO\_DATE(t.BIRTHDATE, 'DD-MON-RR')) = 08 THEN**

**FNAME:=t.FNAME;**

**LNAME:=t.LNAME;**

**BD:=t.BIRTHDATE;**

**END IF;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('THE FIRST AUGUST BORN EMPLOYEE IN THE EMPLOYEE SCHEMA WHEN TRAVERSED IN REVERSE ORDER IS IS'||' '||FNAME||' '||LNAME||' WITH BIRTHDATE'||' '||BD);**

**END;**

**/**

THE FIRST AUGUST BORN EMPLOYEE IN THE EMPLOYEE SCHEMA WHEN TRAVERSED IN REVERSE ORDER IS Larry Gomes WITH BIRTHDATE 28-AUG-96

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 08 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write 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 OR REPLACE PROCEDURE LIST\_STAFF\_DETAILS**

**(V\_SID IN STAFF.SID%TYPE,STAFF\_DETAILS OUT VARCHAR2) AS**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('DETAILS OF THE STAFF ARE:');**

**SELECT SID||' '||NAME||' '||BRANCH||' '||DESG||' '||JOIN\_DT INTO STAFF\_DETAILS**

**FROM STAFF**

**WHERE SID=V\_SID;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('STAFF DOES NOT EXIST...');**

**END;**

**/**

Procedure created.

**DECLARE**

**V\_SID NUMBER := &STAFFID;**

**STAFF\_DETAILS VARCHAR2(100);**

**BEGIN**

**LIST\_STAFF\_DETAILS(V\_SID,STAFF\_DETAILS);**

**DBMS\_OUTPUT.PUT\_LINE(STAFF\_DETAILS);**

**END;**

**/**

Enter value for staffid: 112

old 2: V\_SID NUMBER := &STAFFID;

new 2: V\_SID NUMBER := 112;

PL/SQL procedure successfully completed.

**/**

Enter value for staffid: 113

old 2: V\_SID NUMBER := &STAFFID;

new 2: V\_SID NUMBER := 113;

DETAILS OF THE STAFF ARE:

STAFF DOES NOT EXIST...

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 09 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write 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, maxpay]. If the salary is less than minpay, BELOW\_PAY\_RANGE exception is raised and when cached an appropriate message - "<EmpNo> Receives Salary Below Scale [minpay, maxpay]' is displayed; otherwise ABOVE\_PAY\_RANGE exception is raised and cached to display the 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. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CREATE TABLE PAYSCALE (**

**DESIGNATION VARCHAR2(15) NOT NULL,**

**MINPAY NUMBER(6) NOT NULL,**

**MAXPAY NUMBER(6) NOT NULL,**

**CONSTRAINTS PAYSCALE\_PK\_DESIGNATION PRIMARY KEY (DESIGNATION),**

**CONSTRAINTS PAYSCALE\_CK\_DESIGNATION CHECK**

**(DESIGNATION IN ('Professor','Research Asst.','Asso. Professor', 'Teaching Asst.','Asst. Professor'))**

**);**

Table created.

**DESC PAYSCALE;**

Name Null? Type

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

DESIGNATION NOT NULL VARCHAR2(15)

MINPAY NOT NULL NUMBER(5)

MAXPAY NOT NULL NUMBER(5)

**SELECT TABLE\_NAME,CONSTRAINT\_NAME,CONSTRAINT\_TYPE**

**FROM USER\_CONSTRAINTS**

**WHERE TABLE\_NAME = 'PAYSCALE';**

TABLE\_NAME CONSTRAINT\_NAME C

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

PAYSCALE SYS\_C0012011 C

PAYSCALE SYS\_C0012012 C

PAYSCALE SYS\_C0012013 C

PAYSCALE PAYSCALE\_CK\_DESIGNATION C

PAYSCALE PAYSCALE\_PK\_DESIGNATION P

**SELECT \* FROM PAYSCALE;**

no rows selected

**INSERT INTO PAYSCALE VALUES ('Professor',140000,200000);**

1 row created.

**INSERT INTO PAYSCALE VALUES ('Asso. Professor',100000,140000);**

1 row created**.**

**INSERT INTO PAYSCALE VALUES ('Asst. Professor',50000,90000);**

1 row created.

**INSERT INTO PAYSCALE VALUES ('Teaching Asst.',20000,32500);**

1 row created.

**INSERT INTO PAYSCALE VALUES ('Research Asst.',30000,45000);**

1 row created.

**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

**DECLARE**

**E\_NO EMPLOYEE.ENO%TYPE := &EMPLOYEENUMBER;**

**BELOW\_PAY\_RANGE EXCEPTION;**

**ABOVE\_PAY\_RANGE EXCEPTION;**

**SAL EMPLOYEE.SALARY%TYPE;**

**MIN\_PAY PAYSCALE.MINPAY%TYPE;**

**MAX\_PAY PAYSCALE.MAXPAY%TYPE;**

**BEGIN**

**SELECT SALARY,MINPAY,MAXPAY INTO SAL,MIN\_PAY,MAX\_PAY**

**FROM EMPLOYEE,PAYSCALE**

**WHERE ENO=E\_NO AND EMPLOYEE.DESIGNATION=PAYSCALE.DESIGNATION;**

**IF SAL < MIN\_PAY THEN**

**RAISE BELOW\_PAY\_RANGE;**

**ELSIF SAL > MAX\_PAY THEN**

**RAISE ABOVE\_PAY\_RANGE;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE\_NO:'||E\_NO||' '||'SALARY:'||SAL);**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE DOES NOT EXIST...');**

**WHEN BELOW\_PAY\_RANGE THEN**

**DBMS\_OUTPUT.PUT\_LINE(E\_NO||' Receives Salary Below Scale ['||MIN\_PAY||','||MAX\_PAY||']');**

**WHEN ABOVE\_PAY\_RANGE THEN**

**DBMS\_OUTPUT.PUT\_LINE(E\_NO||' Receives Salary Above Scale ['||MIN\_PAY||','||MAX\_PAY||']');**

**END;**

**/**

Enter value for employeenumber: 7102

old 2: E\_NO EMPLOYEE.ENO%TYPE := &EMPLOYEENUMBER;

new 2: E\_NO EMPLOYEE.ENO%TYPE := 7102;

EMPLOYEE\_NO:7102 SALARY:146500

PL/SQL procedure successfully completed.

**/**

Enter value for employeenumber: 7104

old 2: E\_NO EMPLOYEE.ENO%TYPE := &EMPLOYEENUMBER;

new 2: E\_NO EMPLOYEE.ENO%TYPE := 7104;

7104 Receives Salary Below Scale [140000,200000]

PL/SQL procedure successfully completed.

**/**

Enter value for employeenumber: 7111

old 2: E\_NO EMPLOYEE.ENO%TYPE := &EMPLOYEENUMBER;

new 2: E\_NO EMPLOYEE.ENO%TYPE := 7111;

7111 Receives Salary Above Scale [30000,45000]

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**

**E\_NO EMPLOYEE.ENO%TYPE := &EMPLOYEENUMBER;**

**BELOW\_PAY\_RANGE EXCEPTION;**

**ABOVE\_PAY\_RANGE EXCEPTION;**

**SAL EMPLOYEE.SALARY%TYPE;**

**DESG EMPLOYEE.DESIGNATION%TYPE;**

**MIN\_PAY PAYSCALE.MINPAY%TYPE;**

**MAX\_PAY PAYSCALE.MAXPAY%TYPE;**

**BEGIN**

**FOR REC IN(SELECT ENO AS E\_NO, DESIGNATION AS DESG, SALARY AS SAL,**

**MINPAY AS MIN\_PAY, MAXPAY AS MAX\_PAY**

**FROM EMPLOYEE NATURAL JOIN PAYSCALE) LOOP**

**SAL := REC.SAL;**

**DESG := REC.DESG;**

**MIN\_PAY := REC.MIN\_PAY;**

**MAX\_PAY := REC. MAX\_PAY;**

**E\_NO := REC.E\_NO;**

**BEGIN**

**IF SAL<MIN\_PAY THEN**

**RAISE BELOW\_PAY\_RANGE;**

**ELSIF ESAL>MAX\_PAY THEN**

**RAISE ABOVE\_PAY\_RANGE;**

**END IF;**

**EXCEPTION**

**WHEN BELOW\_PAY\_RANGE THEN**

**DBMS\_OUTPUT.PUT\_LINE (E\_NO||' Receives Salary Below Scale**

**'||MIN\_PAY);**

**WHEN ABOVE\_PAY\_RANGE THEN**

**DBMS\_OUTPUT.PUT\_LINE (E\_NO||' Receives Salary Below Scale**

**'||MAX\_PAY);**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE (E\_NO||' IS INVALID ID’);**

**END;**

**END LOOP;**

**END;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VIVA-VOCE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Q1 – What is an anonymous block?**

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 anonymous block statement, which does not persist in the database, can consist of up to three sections: an optional declaration section, a mandatory executable section, and an optional exception section.

The optional declaration section, which can contain the declaration of variables, cursors, and types that are to be used by statements within the executable and exception sections, is inserted before the executable BEGIN-END block.

The optional exception section can be inserted near the end of the BEGIN-END block. The exception section must begin with the keyword EXCEPTION, and continues until the end of the block in which it appears.

**Q2 – What is an exception? List the standard PL/SQL exceptions.**

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions

* **System-defined exceptions**

These exceptions are predefined in PL/SQL which get raised WHEN certain **database rule is violated.**

* **User-defined exceptions**

This type of users can create their own exceptions according to the need and to raise these exceptions explicitly **raise** command is used.

**Q3 – Differentiate between '&1 and '&&' in SQL.**

There is only one difference between AND and && is that AND is a standard while && is ownership syntax.

Except the above statement, there is no difference between AND and &&.

**Q4 – Why it is a good practice to use %TYPE when declaring variables?**

It is rare that you need to write type annotations in practice. If you provide an initial value for a constant or variable at the point that it is defined, Swift can almost always infer the type to be used for that constant or variable, as described in [Type Safety and Type Inference](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID322)

**INFERENCE:** From this expt, we learnt about anonymous PL/SQL blocks, their

creation, their uses, their implementation and we also executed the

queries regarding to it. We also learnt some basics of procedures.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***