**Experiment No: 05**

**Aim:**

To write and execute PL/SQL blocks (with exception handling) and Cursor

using Oracle 11g.

**Problem Statement:**

Establish the database relation EMPLOYEE and populate it with sample

records. The logical schema of EMPLOYEE table is:

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

**Author : Pravesh Dholwani**

**Roll No. : 108[5A]**

**Date : 16-SEP-21**

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

**EMPLOYEE**(EID,FNAME,LNAME,BIRTHDATE,GENDER,SSN,HIREDATE,SALARY,

DEPARTMENT,DESIGNATION)

**Queries:**

==================

/\*

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 prior to 5 days from the 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

\*/

1. 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 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.

\*/

1. 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 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 and constraints.

\*/

1. 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.
2. 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(), while including proper headers in the result.
3. 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.

Re-execute the block with appropriate modification to test the exception.

1. 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.
2. Write 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 table is searched in reversed order.
3. 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 table PAYSCALE, that includes fields - DESIGNATION (15 alphanumeric: characters), MINPAY (5 digits), MAXPAY (5 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.

\*/

1. 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.

1. 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.
2. Write a SQL code to compile and execute an anonymous block which declares a cursor - FACULTY. The cursor buffers the records comprising - Employee ID, Employee Name (FNAME and LNAME combined) and Designation for the Designation entered by the user.

You may use either EMPLOYEE table or EMPP table for this cursor and print the buffered records. Use %NOTFOUND variable to enable cursor exit

Enter value for faculty designation: LAMBDA

old 5: WHERE UPPER (DESIGNATION) LIKE UPPER('&Faculty Designation%');

new 5: WHERE UPPER(DESIGNATION) LIKE UPPER ('LAMBDAX");

NO MATCHING ROMS FETCHED...

PL/SQL procedure successfully completed.

Enter value for faculty designation: Professor

old 5: WHERE UPPER (DESIGNATION) LIKE UPPER("&Faculty Designation%');

new S: WHERE UPPER(DESIGNATION) LIKE UPPER("Professor%');

7102 Samantha Jones Professor

7101 Eugene Sabatini Professor

7103 Alexander Lloyd Professor

7104 Simon Downing Professor

ALL CURSOR ROWS FETCHED....

PL/SQL procedure successfully completed.

1. CURSOR FOR LOOP:

Modify the cursor in Query-01 as FACULTY CFL which uses the cursor FOR loop to buffering and displaying the records (as mentioned) when employee designation is entered by the user.

Use a variation of cursor FOR loop to include the ROWCOUNT variable to print serial number for the displayed records.

Enter value for faculty designated professor.

old s: WHERE UPPER(DESIGNATIO) LIKE UPPER(‘&Faculty\_Designation%’);

new s: WHERE UPPER(DESIGNATION) LIKE UPPER(‘proFEssor%’);

The Cursor FOR Loop ...

7102 Samantha Jones Professor

7101 Eugene Sabatini Professor

7103 Alexander Lloyd Professor

7104 Simon Downing Professor

1. **EXITING A CURSOR AFTER FETCHING SPECIFIED NUMBER OF ROWS**: Modify the cursor FACULTY\_CFL\_A to display only those many records as desired by the user. Use %ROWCOUNT to enable the cursor to ensure this.
2. **PARAMETERIZED CURSOR WITH DEFAULT VALUES**:

Write a SQL code to compile and execute an anonymous block which declares a cursor - EMP\_SAL\_INFO (Salary, Designation). Let the default values for salart and designation be 75000 and "Asst. Professor" respectively.

The cursor buffers the records comprising - Employee ID, Employee Name (FNAME and LNAME combined), Designation and Salary for the Salary and Designation entered by the user. Use EMPLOYEE table for this cursor. Use this cursor to print the buffered records.

1. **BULK COLLECT with CURSORS**:

Write SQL code to compile and execute a procedure - PRINT EMPLOYEE which receives employee salary as input and prints the following particulars - employee number, employee name and salary, for employees whose salary exceeds the inputted salary. You must use a cursor - SAL\_CURSOR, to buffer required result-set for bulk collect. Use TYPE statement to declare and instantiate array variables. You may also try using %ROWCOUNT. Use EMPP table as source. You may also use EMPLOYEE table.

**Outcomes:**

==============================================

CREATE TABLE EXAM(

UROLL NUMBER NOT NULL,

COURSE VARCHAR2(10) DEFAULT 'DBMS' NOT NULL,

EXAMDT DATE DEFAULT (SYSDATE-5) NOT NULL,

CONSTRAINT EXAM\_PK\_UROLL PRIMARY KEY (UROLL),

CONSTRAINT EXAM\_CHK\_UROLL CHECK (UROLL BETWEEN 1001 AND 1099)

);

SELECT CONSTRAINT\_NAME,CONSTRAINT\_TYPE FROM

USER\_CONSTRAINTS WHERE

TABLE\_NAME = 'EXAM';

CONSTRAINT\_NAME C

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

SYS\_C007359 C

SYS\_C007360 C

SYS\_C007361 C

EXAM\_CHK\_UROLL C

EXAM\_PK\_UROLL P

5 rows selected.

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

Query 1: 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 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.

\*/

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

DECLARE

ROLL NUMBER := 1001;

BEGIN

FOR ROLL IN 1001 .. 1005 LOOP

INSERT INTO EXAM(UROLL)

VALUES(ROLL);

END LOOP;

COMMIT;

END;

/

PL/SQL procedure successfully completed.

SELECT UROLL,COURSE,EXAMDT

FROM EXAM;

UROLL COURSE EXAMDT

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

1001 DBMS 11-SEP-21

1002 DBMS 11-SEP-21

1003 DBMS 11-SEP-21

1004 DBMS 11-SEP-21

1005 DBMS 11-SEP-21

5 rows selected.

CREATE TABLE EMPP(

EID NUMBER(4) NOT NULL,

ENAME VARCHAR2(25) NOT NULL,

HIREDATE DATE NOT NULL,

DESIGNATION VARCHAR2(15) NOT NULL,

SALARY NUMBER(8,2) NOT NULL,

CONSTRAINT EMPP\_PK\_EID PRIMARY KEY(EID)

);

Table created.

SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE

FROM USER\_CONSTRAINTS

WHERE TABLE\_NAME = 'EMPP';

CONSTRAINT\_NAME C

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

SYS\_C007364 C

SYS\_C007365 C

SYS\_C007366 C

SYS\_C007367 C

SYS\_C007368 C

EMPP\_PK\_EID P

6 rows selected.

SELECT \* FROM EMPP;

no rows selected

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

Query 2: 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 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 Stables. Enforce entity integrity constraints on combination of Staff Number and Roll Number. Verify table creation, contents and constraints.

\*/

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

DECLARE

eid EMPLOYEE.ENO%TYPE;

efname EMPLOYEE.FNAME%TYPE;

elname EMPLOYEE.LNAME%TYPE;

ehire\_date EMPLOYEE.HIREDATE%TYPE;

edesignation EMPLOYEE.DESIGNATION%TYPE;

esalary EMPLOYEE.SALARY%TYPE;

e\_count NUMBER ;

BEGIN

SELECT COUNT(\*) into e\_count FROM EMPLOYEE;

DBMS\_OUTPUT.PUT\_LINE(e\_count);

FOR ROW\_NUM IN 1..e\_count LOOP

SELECT ENO,FNAME,LNAME,HIREDATE,DESIGNATION,SALARY

INTO eid,efname,elname,ehire\_date,edesignation,esalary

FROM

(SELECT

rownum as rn,ENO,FNAME,LNAME,HIREDATE,DESIGNATION,SALARY

from employee)

WHERE rn=ROW\_NUM;

INSERT INTO EMPP(EID,ENAME,HIREDATE,DESIGNATION,SALARY)

VALUES(eid,efname||' '||elname,

ehire\_date,edesignation,esalary);

END LOOP;

END;

/

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.

CREATE TABLE MENTEE(

STAFFNO NUMBER(3) NOT NULL,

STAFFNAME VARCHAR2(25) NOT NULL,

STUDENTNAME VARCHAR2(32) NOT NULL,

ROLL NUMBER(3) NOT NULL,

REG\_DT DATE NOT NULL,

CONSTRAINT MENTEE\_PK\_STAFFNOROLL PRIMARY KEY(STAFFNO, ROLL)

);

Table created.

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

Query 3: 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.

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

DECLARE

MENTEE\_REC MENTEE%ROWTYPE;

ROW\_COUNT NUMBER;

BEGIN

SELECT COUNT(\*) INTO ROW\_COUNT

FROM STUDENT

INNER JOIN

STAFF

ON SID=ADVISOR;

FOR KNT IN 1..ROW\_COUNT LOOP

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

INTO MENTEE\_REC.ROLL,MENTEE\_REC.STUDENTNAME,

MENTEE\_REC.REG\_DT,MENTEE\_REC.STAFFNO,MENTEE\_REC.STAFFNAME

FROM

(SELECT ROWNUM AS RN,ROLL,

FNAME,LNAME,REG\_DT,SID,NAME

FROM

(

SELECT ROLL,FNAME,LNAME,REG\_DT,SID,NAME

FROM STUDENT INNER JOIN STAFF ON SID=ADVISOR

)

)

WHERE RN=KNT;

INSERT INTO MENTEE(

STAFFNO,STAFFNAME,STUDENTNAME,ROLL,REG\_DT)

VALUES(

MENTEE\_REC.STAFFNO,MENTEE\_REC.STAFFNAME,

MENTEE\_REC.STUDENTNAME,MENTEE\_REC.ROLL,

MENTEE\_REC.REG\_DT);

END LOOP;

END;

/

PL/SQL procedure successfully completed.

SELECT \* FROM MENTEE;

STAFFNO STAFFNAME STUDENTNAME ROLL REG\_DT

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

101 Kamalkant Marathe Afra Sayed 1 20-JUL-18

104 Aasawari Deodhar Akansha Wasalu 2 20-JUL-18

108 Jasmine Arora Anjali Rajendran 3 19-JUL-18

109 Vallabh Pai Aradhita Menghal 4 07-JUL-18

101 Kamalkant Marathe Ritul Deshmukh 11 18-JUL-18

104 Aasawari Deodhar Sakshi Nema 12 07-JUL-18

108 Jasmine Arora Shreya Agnihotri 13 07-JUL-18

109 Vallabh Pai Shrishti Shukla 14 19-JUL-18

101 Kamalkant Marathe Aayush Muley 31 19-JUL-18

104 Aasawari Deodhar Abhishek Chohan 32 07-JUL-18

108 Jasmine Arora Adesh Kotgirwar 33 20-JUL-18

109 Vallabh Pai Adhney Nawghare 34 08-AUG-18

101 Kamalkant Marathe Ayush Gupta 41 12-JUL-18

104 Aasawari Deodhar Chaitanya Kapre 42 25-JUL-18

108 Jasmine Arora Dev Paliwal 43 21-JUL-18

109 Vallabh Pai Gaurav Shukla 44 17-JUL-18

109 Vallabh Pai Keshubh Sharma 53 20-JUL-18

108 Jasmine Arora Kunal Thorane 54 08-AUG-18

104 Aasawari Deodhar Mehul Khandhadiya 55 19-JUL-18

101 Kamalkant Marathe Nikhil Tiwari 56 04-JUL-18

104 Aasawari Deodhar Rishikesh Kale 63 07-JUL-18

108 Jasmine Arora Ritik Parashar 64 19-JUL-18

STAFFNO STAFFNAME STUDENTNAME ROLL REG\_DT

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

101 Kamalkant Marathe Rohit Chandani 65 08-AUG-18

109 Vallabh Pai Shubham Jha 78 12-JUL-18

108 Jasmine Arora Yaman Kushwah 79 17-JUL-18

104 Aasawari Deodhar Yash Bhageriya 80 19-JUL-18

109 Vallabh Pai Renuka Soni 30 25-JUL-16

108 Jasmine Arora Mayank Rangari 87 25-JUL-16

102 Adishesh Vidyarthi Ketki Fadnavis 5 14-JUL-18

110 Harmeet Khullar Lalita Sharma 6 10-JUL-18

102 Adishesh Vidyarthi Simran Baheti 15 20-JUL-18

110 Harmeet Khullar Urvi Negi 16 19-JUL-18

102 Adishesh Vidyarthi Akshat Chandak 35 20-JUL-18

110 Harmeet Khullar Amey Chole 36 08-AUG-18

110 Harmeet Khullar Gursewak Virdi 45 07-JUL-18

102 Adishesh Vidyarthi Saurabh Khandagale 46 10-AUG-19

102 Adishesh Vidyarthi Paritosh Dandekar 57 14-JUL-18

110 Harmeet Khullar Pavankumar Gupta 58 03-JUL-18

110 Harmeet Khullar Rushil Parikh 71 07-JUL-18

102 Adishesh Vidyarthi Sankalp Pandey 72 07-JUL-18

102 Adishesh Vidyarthi Yash Daware 81 20-JUL-18

110 Harmeet Khullar Yash Roy 82 07-JUL-18

110 Harmeet Khullar Love Sharnagat 68 25-JUL-17

103 Manishi Singh Muskan Gupta 7 19-JUL-18

STAFFNO STAFFNAME STUDENTNAME ROLL REG\_DT

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

106 Deo Narayan Mishra Prateeksha Devikar 8 13-JUL-18

106 Deo Narayan Mishra Deepali Pathe 17 10-AUG-19

103 Manishi Singh Prachi Bhanuse 18 11-AUG-19

103 Manishi Singh Amit Ray 37 20-JUL-18

106 Deo Narayan Mishra Aryan Pandharipande 38 07-JUL-18

106 Deo Narayan Mishra Ganesh Thakur 47 22-AUG-19

103 Manishi Singh Manishkumar Pardhi 48 23-AUG-19

103 Manishi Singh Rahul Agrawal 59 16-JUL-18

106 Deo Narayan Mishra Rajat Chandak 60 20-JUL-18

103 Manishi Singh Saurabh Sushir 73 07-JUL-18

106 Deo Narayan Mishra Shardul Nimbalkar 74 28-JUL-17

106 Deo Narayan Mishra Yash Dhamecha 83 21-JUL-18

103 Manishi Singh Yash Jain 84 03-JUL-18

103 Manishi Singh Anujesh Soni 67 25-JUL-17

105 Geetika Goenka Priyal Taori 9 19-JUL-18

107 Sanjeev Bamireddy Rashi Chouksey 10 08-AUG-18

107 Sanjeev Bamireddy Siddhi Tripathi 19 31-AUG-19

105 Geetika Goenka Atharva Uplanchiwar 39 07-JUL-18

107 Sanjeev Bamireddy Atharva Paliwal 40 20-JUL-18

105 Geetika Goenka Harsh Karwa 51 11-JUL-18

107 Sanjeev Bamireddy Jayesh Kapse 52 08-AUG-18

107 Sanjeev Bamireddy Ram Agrawal 61 19-JUL-18

STAFFNO STAFFNAME STUDENTNAME ROLL REG\_DT

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

105 Geetika Goenka Raunak Khandelwal 62 19-JUL-18

105 Geetika Goenka Shashank Tapas 75 07-JUL-18

107 Sanjeev Bamireddy Shivam Bagadia 76 20-JUL-18

105 Geetika Goenka Shreyas Nemani 77 20-JUL-18

105 Geetika Goenka Yogesh Siral 85 21-JUL-18

107 Sanjeev Bamireddy Shapath Pandey 86 27-JUL-17

107 Sanjeev Bamireddy Ayush Singh 66 27-JUL-17

109 Vallabh Pai Naveen Namjoshi 88 14-AUG-19

110 Harmeet Khullar Tushar Tipnis 89 14-AUG-19

75 rows selected.

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

Query 4: 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(), while including proper headers in the result.

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

BEGIN

DBMS\_OUTPUT.PUT\_LINE(LPAD('STAFFNO',7)||' '

||RPAD('STAFFNAME',20)||' '

||RPAD('STUDENTNAME',20)||' '

||LPAD('ROLL',4)||' '

||RPAD('REG\_DT',8)

);

DBMS\_OUTPUT.PUT\_LINE('-------'||' '

||'--------------------'||' '

||'--------------------'||' '

||'----'||' '

||'--------'

);

FOR T IN (SELECT \* FROM MENTEE) LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(T.STAFFNO,7)||' '

||RPAD(T.STAFFNAME,20)||' '

||RPAD(T.STUDENTNAME,20)||' '

||LPAD(T.ROLL,4)||' '

||RPAD(T.REG\_DT,8)

);

END LOOP;

END;

/

STAFFNO STAFFNAME STUDENTNAME ROLL REG\_DT

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

101 Kamalkant Marathe Afra Sayed 1 20-JUL-1

104 Aasawari Deodhar Akansha Wasalu 2 20-JUL-1

108 Jasmine Arora Anjali Rajendran 3 19-JUL-1

109 Vallabh Pai Aradhita Menghal 4 07-JUL-1

101 Kamalkant Marathe Ritul Deshmukh 11 18-JUL-1

104 Aasawari Deodhar Sakshi Nema 12 07-JUL-1

108 Jasmine Arora Shreya Agnihotri 13 07-JUL-1

109 Vallabh Pai Shrishti Shukla 14 19-JUL-1

101 Kamalkant Marathe Aayush Muley 31 19-JUL-1

104 Aasawari Deodhar Abhishek Chohan 32 07-JUL-1

108 Jasmine Arora Adesh Kotgirwar 33 20-JUL-1

109 Vallabh Pai Adhney Nawghare 34 08-AUG-1

101 Kamalkant Marathe Ayush Gupta 41 12-JUL-1

104 Aasawari Deodhar Chaitanya Kapre 42 25-JUL-1

108 Jasmine Arora Dev Paliwal 43 21-JUL-1

109 Vallabh Pai Gaurav Shukla 44 17-JUL-1

109 Vallabh Pai Keshubh Sharma 53 20-JUL-1

108 Jasmine Arora Kunal Thorane 54 08-AUG-1

104 Aasawari Deodhar Mehul Khandhadiya 55 19-JUL-1

101 Kamalkant Marathe Nikhil Tiwari 56 04-JUL-1

104 Aasawari Deodhar Rishikesh Kale 63 07-JUL-1

108 Jasmine Arora Ritik Parashar 64 19-JUL-1

101 Kamalkant Marathe Rohit Chandani 65 08-AUG-1

109 Vallabh Pai Shubham Jha 78 12-JUL-1

108 Jasmine Arora Yaman Kushwah 79 17-JUL-1

104 Aasawari Deodhar Yash Bhageriya 80 19-JUL-1

109 Vallabh Pai Renuka Soni 30 25-JUL-1

108 Jasmine Arora Mayank Rangari 87 25-JUL-1

102 Adishesh Vidyarthi Ketki Fadnavis 5 14-JUL-1

110 Harmeet Khullar Lalita Sharma 6 10-JUL-1

102 Adishesh Vidyarthi Simran Baheti 15 20-JUL-1

110 Harmeet Khullar Urvi Negi 16 19-JUL-1

102 Adishesh Vidyarthi Akshat Chandak 35 20-JUL-1

110 Harmeet Khullar Amey Chole 36 08-AUG-1

110 Harmeet Khullar Gursewak Virdi 45 07-JUL-1

102 Adishesh Vidyarthi Saurabh Khandagale 46 10-AUG-1

102 Adishesh Vidyarthi Paritosh Dandekar 57 14-JUL-1

110 Harmeet Khullar Pavankumar Gupta 58 03-JUL-1

110 Harmeet Khullar Rushil Parikh 71 07-JUL-1

102 Adishesh Vidyarthi Sankalp Pandey 72 07-JUL-1

102 Adishesh Vidyarthi Yash Daware 81 20-JUL-1

110 Harmeet Khullar Yash Roy 82 07-JUL-1

110 Harmeet Khullar Love Sharnagat 68 25-JUL-1

103 Manishi Singh Muskan Gupta 7 19-JUL-1

106 Deo Narayan Mishra Prateeksha Devikar 8 13-JUL-1

106 Deo Narayan Mishra Deepali Pathe 17 10-AUG-1

103 Manishi Singh Prachi Bhanuse 18 11-AUG-1

103 Manishi Singh Amit Ray 37 20-JUL-1

106 Deo Narayan Mishra Aryan Pandharipande 38 07-JUL-1

106 Deo Narayan Mishra Ganesh Thakur 47 22-AUG-1

103 Manishi Singh Manishkumar Pardhi 48 23-AUG-1

103 Manishi Singh Rahul Agrawal 59 16-JUL-1

106 Deo Narayan Mishra Rajat Chandak 60 20-JUL-1

103 Manishi Singh Saurabh Sushir 73 07-JUL-1

106 Deo Narayan Mishra Shardul Nimbalkar 74 28-JUL-1

106 Deo Narayan Mishra Yash Dhamecha 83 21-JUL-1

103 Manishi Singh Yash Jain 84 03-JUL-1

103 Manishi Singh Anujesh Soni 67 25-JUL-1

105 Geetika Goenka Priyal Taori 9 19-JUL-1

107 Sanjeev Bamireddy Rashi Chouksey 10 08-AUG-1

107 Sanjeev Bamireddy Siddhi Tripathi 19 31-AUG-1

105 Geetika Goenka Atharva Uplanchiwar 39 07-JUL-1

107 Sanjeev Bamireddy Atharva Paliwal 40 20-JUL-1

105 Geetika Goenka Harsh Karwa 51 11-JUL-1

107 Sanjeev Bamireddy Jayesh Kapse 52 08-AUG-1

107 Sanjeev Bamireddy Ram Agrawal 61 19-JUL-1

105 Geetika Goenka Raunak Khandelwal 62 19-JUL-1

105 Geetika Goenka Shashank Tapas 75 07-JUL-1

107 Sanjeev Bamireddy Shivam Bagadia 76 20-JUL-1

105 Geetika Goenka Shreyas Nemani 77 20-JUL-1

105 Geetika Goenka Yogesh Siral 85 21-JUL-1

107 Sanjeev Bamireddy Shapath Pandey 86 27-JUL-1

107 Sanjeev Bamireddy Ayush Singh 66 27-JUL-1

109 Vallabh Pai Naveen Namjoshi 88 14-AUG-1

110 Harmeet Khullar Tushar Tipnis 89 14-AUG-1

PL/SQL procedure successfully completed.

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

Query 5: 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.

Re-execute the block with appropriate modification to test the exception.

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

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

Query 6: 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

DBMS\_OUTPUT.PUT\_LINE(RPAD('ENO',5)||' '

||RPAD('NAME',25));

DBMS\_OUTPUT.PUT\_LINE('-----'||' '

||'-------------------------');

FOR T IN (SELECT ENO,FNAME,LNAME,

(TO\_DATE(CURRENT\_DATE,'DD-MM-YY')-HIREDATE)/360 AS YEARS

FROM EMPLOYEE) LOOP

IF T.YEARS>=12.0 THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD(T.ENO,5)||' '

||RPAD(T.FNAME||' '||T.LNAME,25));

END IF;

END LOOP;

END;

/

ENO NAME

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

7102 Samantha Jones

7101 Eugene Sabatini

7103 Alexander Lloyd

7104 Simon Downing

7107 Christov Plutnik

7105 Christina Mulboro

7106 Dolly Silverline

7108 Ellena Sanchez

7109 Martina Jacobson

PL/SQL procedure successfully completed.

BEGIN

DBMS\_OUTPUT.PUT\_LINE(RPAD('ENO',5)||' '

||RPAD('NAME',25));

DBMS\_OUTPUT.PUT\_LINE('-----'||' '

||'-------------------------');

FOR T IN (SELECT ENO,FNAME,LNAME,

(TO\_DATE(CURRENT\_DATE,'DD-MM-YY')-HIREDATE)/360 AS YEARS

FROM EMPLOYEE) LOOP

IF T.YEARS>=10.0 THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD(T.ENO,5)||' '

||RPAD(T.FNAME||' '||T.LNAME,25));

END IF;

END LOOP;

END;

/

ENO NAME

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

7102 Samantha Jones

7101 Eugene Sabatini

7103 Alexander Lloyd

7104 Simon Downing

7107 Christov Plutnik

7105 Christina Mulboro

7106 Dolly Silverline

7108 Ellena Sanchez

7109 Martina Jacobson

7110 William Smithfield

PL/SQL procedure successfully completed.

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

Write 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

table is searched in reversed order.

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

DECLARE

EMP\_ROW EMPLOYEE%ROWTYPE;

BEGIN

FOR EMP\_ROW IN (SELECT \* FROM EMPLOYEE)

LOOP

IF EXTRACT(MONTH FROM EMP\_ROW.BIRTHDATE)=8 THEN

DBMS\_OUTPUT.PUT\_LINE(EMP\_ROW.ENO||' '||EMP\_ROW.FNAME

||' '||EMP\_ROW.LNAME);

END IF;

EXIT WHEN EXTRACT(MONTH FROM EMP\_ROW.BIRTHDATE)=8;

END LOOP;

END;

/

7114 Larry Gomes

PL/SQL procedure successfully completed.

DECLARE

EMP\_ROW EMPLOYEE%ROWTYPE;

BEGIN

FOR EMP\_ROW IN (SELECT \* FROM EMPLOYEE

ORDER BY ENO DESC)

LOOP

IF EXTRACT(MONTH FROM EMP\_ROW.BIRTHDATE)=8 THEN

DBMS\_OUTPUT.PUT\_LINE(EMP\_ROW.ENO||' '||EMP\_ROW.FNAME||' '||EMP\_ROW.LNAME);

END IF;

EXIT WHEN EXTRACT(MONTH FROM EMP\_ROW.BIRTHDATE)=8;

END LOOP;

END;

/

7114 Larry Gomes

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.

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

DECLARE

STAFF\_ROW STAFF%ROWTYPE;

ID NUMBER;

BEGIN

ID:='&STAFF\_ID';

SELECT \* INTO STAFF\_ROW

FROM STAFF WHERE

SID=ID;

DBMS\_OUTPUT.PUTLINE(STAFF\_ROW.SID||' '||STAFF\_ROW.NAME||' '||

STAFF\_ROW.BRANCH||' '||STAFF\_ROW.DESG||' '||STAFF\_ROW.JOIN\_DT);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO RECORD FOUND');

END;

/

Enter value for staff\_id: 101

old 5: ID:='&STAFF\_ID';

new 5: ID:='101';

101 KamalkantMarathe CSE Professor 12-JUN-05

PL/SQL procedure successfully completed.

Enter value for staff\_id: 119

old 5: ID:='&STAFF\_ID';

new 5: ID:='119';

NO RECORD FOUND

PL/SQL procedure successfully completed.

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

/\*

Create table PAYSCALE, that includes fields - DESIGNATION (15 alphanumeric:

characters), MINPAY (5 digits), MAXPAY (5 digits). Entity Integrity is

maintained on DESIGNATION, with plossible 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.

\*/

9. 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),

MINPAY NUMBER(6),

MAXPAY NUMBER(6),

CONSTRAINT PAYSCALE\_CK\_DESIG CHECK (DESIGNATION IN

('Professor','ResearchAsst.','Asso. Professor','Teaching Asst.'

,'Asst. Professor')),

CONSTRAINT PAYSCALE\_PK\_DESIG PRIMARY KEY (DESIGNATION)

);

Table created.

INSERT INTO PAYSCALE(DESIGNATION,MINPAY,MAXPAY) VALUES

('Professor',140000,200000);

1 row created.

INSERT INTO PAYSCALE(DESIGNATION,MINPAY,MAXPAY) VALUES

('Asso. Professor',100000,140000

1 row created.

INSERT INTO PAYSCALE(DESIGNATION,MINPAY,MAXPAY) VALUES

('Asst. Professor',50000,90000);

1 row created.

INSERT INTO PAYSCALE(DESIGNATION,MINPAY,MAXPAY) VALUES

('Teaching Asst.',20000,32500);

1 row created.

INSERT INTO PAYSCALE(DESIGNATION,MINPAY,MAXPAY) VALUES

('Research Asst.',30000,45000);

1 row created.

DECLARE

ENO\_INP EMPLOYEE.ENO%TYPE;

P\_MINPAY PAYSCALE.MINPAY%TYPE;

E\_SAL EMPLOYEE.SALARY%TYPE;

BELOW\_PAY\_RANGE EXCEPTION;

ABOVE\_PAY\_RANGE EXCEPTION;

P\_MAXPAY PAYSCALE.MAXPAY%TYPE;

BEGIN

ENO\_INP:='&EMPLOYEE\_NUMBER';

SELECT EMPLOYEE.SALARY,PAYSCALE.MINPAY,PAYSCALE.MAXPAY

INTO E\_SAL,P\_MINPAY,P\_MAXPAY

FROM EMPLOYEE INNER JOIN PAYSCALE

USING (DESIGNATION)

WHERE EMPLOYEE.ENO=ENO\_INP;

IF E\_SAL>P\_MAXPAY THEN

RAISE ABOVE\_PAY\_RANGE;

ELSIF E\_SAL<P\_MINPAY THEN

RAISE BELOW\_PAY\_RANGE;

ELSE

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE'||ENO\_INP||' HAS A SALARY WITHIN PAY RANGE');

END IF;

EXCEPTION

WHEN BELOW\_PAY\_RANGE THEN

DBMS\_OUTPUT.PUT\_LINE(ENO\_INP||' RECEIVES SALARY BELOW SCALE '||'['||P\_MINPAY||','

||P\_MAXPAY||']');

WHEN ABOVE\_PAY\_RANGE THEN

DBMS\_OUTPUT.PUT\_LINE(ENO\_INP||' RECEIVES SALARY ABOVE SCALE '||'['||P\_MINPAY||','

||P\_MAXPAY||']');

END;

/

Enter value for employee\_number: 7102

old 9: ENO\_INP:='&EMPLOYEE\_NUMBER';

new 9: ENO\_INP:='7102';

EMPLOYEE7102 HAS A SALARY WITHIN PAY RANGE

PL/SQL procedure successfully completed.

Enter value for employee\_number: 7104

old 9: ENO\_INP:='&EMPLOYEE\_NUMBER';

new 9: ENO\_INP:='7104';

7104 RECEIVES SALARY BELOW SCALE [140000,200000]

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\_NO EMPLOYEE.ENO%TYPE;

EMP\_SAL EMPLOYEE.SALARY%TYPE;

MIN\_PAY PAYSCALE.MINPAY%TYPE;

MAX\_PAY PAYSCALE.MAXPAY%TYPE;

ABOVE\_PAY\_RANGE EXCEPTION;

BELOW\_PAY\_RANGE EXCEPTION;

BEGIN

FOR I IN (SELECT EMPLOYEE.SALARY AS EMP\_SAL,EMPLOYEE.ENO AS EMP\_NO,

PAYSCALE.MINPAY AS MIN\_PAY,PAYSCALE.MAXPAY AS MAX\_PAY

FROM EMPLOYEE NATURAL JOIN PAYSCALE)

LOOP

BEGIN

IF I.EMP\_SAL > I.MAX\_PAY THEN

RAISE ABOVE\_PAY\_RANGE;

ELSIF I.EMP\_SAL < I.MIN\_PAY THEN

RAISE BELOW\_PAY\_RANGE;

ELSE

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

END IF;

EXCEPTION

WHEN ABOVE\_PAY\_RANGE THEN

DBMS\_OUTPUT.PUT\_LINE(I.EMP\_NO||' Receives

Salary Above Scale '||'['||I.MIN\_PAY||','||

I.MAX\_PAY||']'

);

WHEN BELOW\_PAY\_RANGE THEN

DBMS\_OUTPUT.PUT\_LINE(I.EMP\_NO||' Receives

Salary Below Scale '||'['||I.MIN\_PAY||','||

I.MAX\_PAY||']'

);

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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **QUERY 11** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write a SQL code to compile and execute an anonymous block which declares a

cursor - FACULTY. The cursor buffers the records comprising - Employee ID,

Employee Name (FNAME and LNAME combined) and Designation for the Designation

entered by the user.

You may use either EMPLOYEE table or EMPP table for this cursor and print the

buffered records. Use %NOTFOUND variable to enable cursor exit.

Enter value for faculty\_designation: LAMBDA

old 5: WHERE UPPER(DESIGNATION) LIKE UPPER('&Faculty\_Designation%');

new 5: WHERE UPPER(DESIGNATION) LIKE UPPER('LAMBDA%');

NO MATCHING ROWS FETCHED ..

PL/SQL procedure successfully completed.

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

DECLARE

FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

CURSOR FACULTY IS

SELECT ENO,FNAME||' '||LNAME AS ENAME,

DESIGNATION FROM EMPLOYEE

WHERE UPPER(DESIGNATION) = UPPER(FACULTY\_DESIG);

C\_FAC FACULTY%ROWTYPE;

BEGIN

OPEN FACULTY;

LOOP

FETCH FACULTY INTO C\_FAC;

EXIT WHEN FACULTY%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(C\_FAC.ENO||' '||C\_FAC.ENAME||

' '||C\_FAC.DESIGNATION);

END LOOP;

IF FACULTY%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('NO MATCHING ROWS FETCHED');

END IF;

CLOSE FACULTY;

END;

/

Enter value for enter\_designation: Professor

old 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

new 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := 'Professor';

7102 Samantha Jones Professor

7101 Eugene Sabatini Professor

7103 Alexander Lloyd Professor

7104 Simon Downing Professor

PL/SQL procedure successfully completed.

Enter value for enter\_designation: LAMBDA

old 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

new 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := 'LAMBDA';

NO MATCHING ROWS FETCHED

PL/SQL procedure successfully completed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **QUERY 12** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CURSOR FOR LOOP:**

Modify the cursor in Query-01 as FACULTY CFL which uses the cursor FOR loop to

buffering and displaying the records (as mentioned) when employee designation

is entered by the user.

Use a variation of cursor FOR loop to include the ROWCOUNT variable to print

serial number for the displayed records.

Enter value for faculty designated professor.

old s: WHERE UPPER(DESIGNATIO) LIKE UPPER(‘&Faculty\_Designation%’);

new s: WHERE UPPER(DESIGNATION) LIKE UPPER(‘proFEssor%’);

The Cursor FOR Loop ...

7102 Samantha Jones Professor

7101 Eugene Sabatini Professor

7103 Alexander Lloyd Professor

7104 Simon Downing Professor

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

DECLARE

FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

CURSOR FACULTY\_CFL IS

SELECT ENO,FNAME||' '||LNAME AS ENAME,

DESIGNATION FROM EMPLOYEE

WHERE UPPER(DESIGNATION) = UPPER(FACULTY\_DESIG);

C\_FAC FACULTY\_CFL%ROWTYPE;

CKNT NUMBER:=0;

BEGIN

FOR C\_FAC IN FACULTY\_CFL

LOOP

DBMS\_OUTPUT.PUT\_LINE(C\_FAC.ENO||' '||C\_FAC.ENAME||

' '||C\_FAC.DESIGNATION);

CKNT:=1;

END LOOP;

` IF CKNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('NO MATCHING ROWS FETCHED');

END IF;

END;

/

Enter value for enter\_designation: Asst. Professor

old 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

new 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := 'Asst. Professor';

7109 Martina Jacobson Asst. Professor

7110 William Smithfield Asst. Professor

PL/SQL procedure successfully completed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **QUERY 13** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**EXITING A CURSOR AFTER FETCHING SPECIFIED NUMBER OF ROWS:** Modify the cursor

FACULTY\_CFL\_A to display only those many records as desired by the user. Use

%ROWCOUNT to enable the cursor to ensure this.

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

DECLARE

FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

CURSOR FACULTY\_CFL IS

SELECT ENO,FNAME||' '||LNAME AS ENAME,

DESIGNATION FROM EMPLOYEE

WHERE UPPER(DESIGNATION) = UPPER(FACULTY\_DESIG);

C\_FAC FACULTY\_CFL%ROWTYPE;

COUNT1 NUMBER := 1;

NUM NUMBER := &ENTER\_NUMBER\_OF\_RECORDS;

BEGIN

FOR CC IN FACULTY\_CFL LOOP

DBMS\_OUTPUT.PUT\_LINE(FACULTY\_CFL%ROWCOUNT||' '|| CC.ENO||' '

||CC.ENAME||' '||CC.DESIGNATION);

COUNT1 := COUNT1 + 1;

EXIT WHEN FACULTY\_CFL%ROWCOUNT = NUM;

END LOOP;

IF COUNT1 = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('NO MATCHING ROWS FETCHED');

END IF;

END;

/

Enter value for enter\_designation: Professor

old 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := '&ENTER\_DESIGNATION';

new 2: FACULTY\_DESIG EMPLOYEE.DESIGNATION%TYPE := 'Professor';

Enter value for enter\_number\_of\_records: 2

old 9: NUM NUMBER := &ENTER\_NUMBER\_OF\_RECORDS;

new 9: NUM NUMBER := 2;

1 7102 Samantha Jones Professor

2 7101 Eugene Sabatini Professor

PL/SQL procedure successfully completed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **QUERY 14** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PARAMETERIZED CURSOR WITH DEFAULT VALUES:**

Write a SQL code to compile and execute an anonymous block which declares a

cursor - EMP\_SAL\_INFO (Salary, Designation). Let the default values for salary

and designation be 75000 and "Asst. Professor" respectively.

The cursor buffers the records comprising - Employee ID, Employee Name

(FNAME and LNAME combined), Designation and Salary for the Salary and

Designation entered by the user. Use EMPLOYEE table for this cursor.

Use this cursor to print the buffered records.

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

DECLARE

CURSOR EMP\_SAL\_INFO(SAL EMPLOYEE.SALARY%TYPE := 75000,DESG

EMPLOYEE.DESIGNATION%TYPE := 'Asst. Professor')

IS SELECT ENO, LNAME||' '||FNAME AS NAME,DESIGNATION,SALARY

FROM EMPLOYEE

WHERE UPPER(EMPLOYEE.DESIGNATION)=UPPER(DESG) AND EMPLOYEE.SALARY>SAL;

SPECIFIED\_SALARY NUMBER(6);

SALARY1 NUMBER(6);

SALARY2 NUMBER(6);

SPECIFIED\_DESIGNATION EMPLOYEE.DESIGNATION%TYPE;

EMP\_REC EMP\_SAL\_INFO%ROWTYPE;

BEGIN

SPECIFIED\_SALARY:=&SPECIFIED\_SALARY;

SALARY1:=SPECIFIED\_SALARY;

SPECIFIED\_SALARY:=&SPECIFIED\_SALARY;

SALARY2:=SPECIFIED\_SALARY;

SPECIFIED\_DESIGNATION:='&SPECIFIED\_DESIGNATION';

DBMS\_OUTPUT.PUT\_LINE('WITH DEFAULT VALUES ....');

OPEN EMP\_SAL\_INFO();

LOOP

FETCH EMP\_SAL\_INFO INTO EMP\_REC;

EXIT WHEN EMP\_SAL\_INFO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(EMP\_REC.ENO||CHR(9)||EMP\_REC.NAME||CHR(9)||

EMP\_REC.DESIGNATION||CHR(9) ||EMP\_REC.SALARY);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

CLOSE EMP\_SAL\_INFO;

DBMS\_OUTPUT.PUT\_LINE('WITH SOME DEFAULT VALUES ....');

OPEN EMP\_SAL\_INFO(SALARY1);

LOOP

FETCH EMP\_SAL\_INFO INTO EMP\_REC;

EXIT WHEN EMP\_SAL\_INFO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(EMP\_REC.ENO||CHR(9)||EMP\_REC.NAME||CHR(9)||

EMP\_REC.DESIGNATION||CHR(9) ||EMP\_REC.SALARY);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

CLOSE EMP\_SAL\_INFO;

DBMS\_OUTPUT.PUT\_LINE('WITH ALL APPLIED VALUES ....');

OPEN EMP\_SAL\_INFO(SALARY2,SPECIFIED\_DESIGNATION);

LOOP

FETCH EMP\_SAL\_INFO INTO EMP\_REC;

EXIT WHEN EMP\_SAL\_INFO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(EMP\_REC.ENO||CHR(9)||EMP\_REC.NAME||CHR(9)||

EMP\_REC.DESIGNATION||CHR(9) ||EMP\_REC.SALARY);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

CLOSE EMP\_SAL\_INFO;

DBMS\_OUTPUT.PUT\_LINE('ALL CASES DONE');

END;

/

Enter value for specified\_salary: 88000

old 10: specified\_salary:=&SPECIFIED\_SALARY;

new 10: specified\_salary:=88000;

Enter value for specified\_salary: 120000

old 12: specified\_salary:=&SPECIFIED\_SALARY;

new 12: specified\_salary:=120000;

Enter value for specified\_designation: ASSI. PROFESSOR

old 14: specified\_designation:='&Specified\_Designation';

new 14: specified\_designation:='ASSI. PROFESSOR';

With Default Values ....

7109 Jacobson Martina Asst. Professor 91000

7110 Smithfield William Asst. Professor 86400

With Some Default Values ....

7109 Jacobson Martina Asst. Professor 91000

With all applied values ....

7107 PlutnikChristov Asst. Professor 127400

7105 Mulboro Christina Asst. Professor 127400

7106 Silverline Dolly Asst. Professor 127400

ALL CASES DONE

PL/SQL procedure successfully completed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **QUERY 15** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**BULK COLLECT with CURSORS:**

Write SQL code to compile and execute a procedure - PRINT EMPLOYEE which

receives employee salary as input and prints the following particulars -

employee number, employee name and salary, for employees whose salary exceeds

the inputted salary. You must use a cursor - SAL\_CURSOR, to buffer required

result-set for bulk collect. Use TYPE statement to declare and instantiate

array variables. You may also try using %ROWCOUNT. Use EMPP table as source.

You may also use EMPLOYEE table.

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

CREATE OR REPLACE PROCEDURE PRINT\_EMPLOYEE(SAL EMPLOYEE.SALARY%TYPE) AS

CURSOR SAL\_CURSOR IS

SELECT \* FROM EMPLOYEE

WHERE SALARY>PRINT\_EMPLOYEE.SAL;

TYPE SAL\_CURSOR\_TAB IS TABLE OF SAL\_CURSOR%ROWTYPE;

EMP\_REC SAL\_CURSOR\_TAB;

EMP\_ROW EMPLOYEE%ROWTYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEES HAVING SALARY >'||PRINT\_EMPLOYEE.SAL);

DBMS\_OUTPUT.PUT\_LINE(RPAD('ENO',5)||RPAD('NAME',20)||'SALARY');

DBMS\_OUTPUT.PUT\_LINE(RPAD('---',5)||RPAD('--------------',20)||

'--------');

OPEN SAL\_CURSOR;

FETCH SAL\_CURSOR BULK COLLECT INTO EMP\_REC ;

FOR KNT IN EMP\_REC.FIRST .. EMP\_REC.LAST

LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(EMP\_REC(KNT).ENO,5)||

RPAD(EMP\_REC(KNT).FNAME||' '

||EMP\_REC(KNT).LNAME,20)||EMP\_REC(KNT).SALARY);

END LOOP;

END;

/

Procedure created.

EXECUTE PRINT\_EMPLOYEE(125000)

EMPLOYEES HAVING SALARY >125000

ENO NAME SALARY

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

7102 Samantha Jones 146500

7101 Eugene Sabatini 150000

7103 Alexander Lloyd 148000

7104 Simon Downing 138400

7107 ChristovPlutnik 127400

7105 Christina Mulboro 127400

7106 Dolly Silverline 127400

PL/SQL procedure successfully completed.

EXECUTE PRINT\_EMPLOYEE(100000)

EMPLOYEES HAVING SALARY >100000

ENO NAME SALARY

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

7102 Samantha Jones 146500

7101 Eugene Sabatini 150000

7103 Alexander Lloyd 148000

7104 Simon Downing 138400

7107 ChristovPlutnik 127400

7105 Christina Mulboro 127400

7106 Dolly Silverline 127400

7108 Ellena Sanchez 119700

PL/SQL procedure successfully completed.

**VIVA VOCE:**

**==========**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -01** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What is an anonymous block?

**ANS.**

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. In PL/SQL contexts,

this statement can be compiled and executed by the data server.

EXAMPLE:

BEGIN

DBMS\_OUTPUT.PUT\_LINE('HELLO WORLD');

END;

/

HELLO WORLD

PL/SQL procedure successfully completed.

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -02** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**ANS.**

An exception is a PL/SQL error that is raised during program execution,

either implicitly 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:

1. ACCESS\_INTO\_NULL

2. CASE\_NOT\_FOUND

3. COLLECTION\_IS\_NULL

4. CURSOR\_ALREADY\_OPEN

5. DUP\_VAL\_ON\_INDEX

6. INVALID\_CURSOR

7. INVALID\_NUMBER

8. LOGIN\_DENIED

9. NO\_DATA\_FOUND

10. TOO\_MANY\_RESOURCE

11. VALUE\_ERROR

12. ZERO\_DIVIDE

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -03** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Differentiate between '&' and '&&' in SQL.

ANS.

"&” is used to create a temporary substitution variable. You will be

prompted to enter the value every time the variable is referenced.

“&&” is used to create a permanent substitution variable. You need to

enter the value only once.

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -04** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Why it is good practice to use %TYPE when declaring variables?

**ANS.**

The %TYPE attribute, used in PL/SQL variable and parameter declarations,

is supported by the data server. Use of this attribute ensures that type

compatibility between table columns and PL/SQL variables is maintained.

A qualified column name in dot notation or the name of a previously

declared variable must be specified as a prefix to the %TYPE attribute.

The data type of this column or variable is assigned to the variable being

declared. If the data type of the column or variable changes, there is no

need to modify the declaration code.

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -05** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What is cursor?List the steps associated with implementing a cursor.

**ANS.**

A cursor is a pointer to this context area. PL/SQL controls the context

area through a cursor. A cursor holds the rows (one or more) returned by a

SQL statement. The set of rows the cursor holds is referred to as the

active set.

Steps associated with implementing a cursor:-

1. Declaring the Cursor

2. Opening the Cursor

3. Fetching the Cursor

4. Closing the Cursor

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -06** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What is "active set"?

**ANS.**

Once a cursor is opened. It stands for some selection of rows. The set of

all rows that are produced by the query of a cursor is called “active set”.

Active set can be thought of as a collection of rows and cursor as the

pointer to one of this rows.

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -07** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What are the advantages of a cursor FOR loop?

**ANS.**

Advantages of Cursor FOR loop:

1.No need to open the cursor.

2.Fetch the records automatically.

3.It automatically checks the end of rows.

4.It automatically closes the cursor.

5.No need to declare the variables.

6.Code size will be decreased.

7.Execution will be faster.

8.Less fetching time.

9.It is collection of information from cursor to a variable.

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **VVQ -08** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Why it is a good practice to close a cursor?

**ANS.**

When a cursor is opened, Oracle runs the query to generate the results and

positions the cursor before the first row of the result set. However, a

cursor can only be opened if it is not already open, attempting to open a

cursor that is already open generates a "CURSOR\_ALREADY\_OPEN" exception.

In other words if you declare a cursor and open it, if you try to open it

again withouth closing it, Oracle raises an exception.Hence it is a good

practice to close a cursor.

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

**INFERENCES:**

**===========**

1. In this practical, we studied the basics of PL/SQL and the structure of

code for creating an PL/SQL block.

2. We also studied the concpts like Exception,Cursors,Cursor FOR Loops and

concept of Bulk Collecting through this practical.

**===============================================================================**