NAME : ANSARI TOSIFHUSAIN TAHIRHUSAIN

DIV : G-05

DAY-1

QUESTION...

Essential List (Practical)

1 Creating Table with Data Constraints:

Viewing data in the tables: (Select Statement and where condition)

Filtering Data from the table

Sorting Data in the Table: (Order By, Desc clause)

Creating a Table from a Table

Inserting Data into a Table from Another Table

Deleting records from the table:

Using Delete and Truncate operations Destroying Table

2 CUST (custno, custname, addln1, addln2, city, state, phone)

ITEM (itemno, itemname, itemprice, qty\_on\_hand)

INVOICE (invno, invDate, custno)

INV\_ITEM (invno, itemno, qty\_used)

ANSWER...

SQL> SELECT \* FROM CUST;

CUSTNO CUSTNAME ADDLN1 ADDLN2

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

CITY STATE

PHONE

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

1 John Doe 123 Main St Apt 1

New York NY

1234567890

2 Jane Smith 456 Elm St Suite 5

Los Angeles CA

0987654321

SQL> SELECT \* FROM ITEM;

ITEMNO ITEMNAME ITEMPRICE QTY\_ON\_HAND

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

1 Laptop 1000 10

2 Mouse 20 100

3 Keyboard 50 50

SQL> SELECT \* FROM INVOICE;

INVNO INVDATE CUSTNO

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

1 16-OCT-24 1

2 17-OCT-24 2

SQL> SELECT \* FROM INV\_ITEM;

INVNO ITEMNO QTY\_USED

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

1 1 1

1 2 2

2 3 1

SQL>

SELECT \* FROM CUST WHERE city = 'New York';

CUSTNO CUSTNAME ADDLN1 ADDLN2

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

CITY STATE

PHONE

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

1 John Doe 123 Main St Apt 1

New York NY

1234567890

SQL> SELECT \* FROM ITEM ORDER BY itemprice DESC;

ITEMNO ITEMNAME ITEMPRICE QTY\_ON\_HAND

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

1 Laptop 1000 10

3 Keyboard 50 50

2 Mouse 20 100

SQL> CREATE TABLE CUST\_BACKUP AS SELECT \* FROM CUST;

Table created.

SQL> INSERT INTO CUST\_BACKUP SELECT \* FROM CUST WHERE state = 'NY';

1 row created.

SQL> DELETE FROM CUST WHERE custno = 2;

DELETE FROM CUST WHERE custno = 2

SQL> TRUNCATE TABLE CUST\_BACKUP;

Table truncated.

SQL> DROP TABLE CUST\_BACKUP;

Table dropped.

Day-2

QUESTION...

Essential List (Practical)

1 1. Create the above four tables along with key constraints. 2. Write an Insert script for

insertion of rows with substitution variables 3. Insert appropriate data

2 Add a column – “color” to the Item table.

1. Display the column Item name and Price in sentence form using Concatenation.

ANSWER...

SQL> ALTER TABLE ITEM

2 ADD color VARCHAR(20);

Table altered.

SQL> SELECT 'The item ' || itemname || ' costs $' || itemprice AS "Item Description"

2 FROM ITEM;

Item Description

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

The item Laptop costs $1000

The item Mouse costs $20

The item Keyboard costs $50

QUESTION...

Desirable List (Practical):

1 . Display all the records of EMP table

Display all the records of DEPT table

Display only Name of all employees

Display Employee’s Name and salary

Display only unique departments (deptno) from EMP table

ANSWER...

SQL> SELECT \* FROM DEPT11;

no rows selected

SQL>

SQL> SELECT ename FROM EMP11;

no rows selected

SQL>

SQL> SELECT ename, sal FROM EMP11;

no rows selected

SQL>

SQL> SELECT DISTINCT deptno FROM EMP11;

no rows selected

DAY-3

QUESTION 1...

Essential List (Practical)

1. Create the above four tables along with key constraints. 2. Write an Insert script for insertion of

rows with substitution variables 3. Insert appropriate data

Add a column – “color” to the Item table.

1. Display the column Item name and Price in sentence form using Concatenation.

ANSWER...

SQL> CREATE TABLE EMP51 (

2 EMP\_ID INT PRIMARY KEY,

3 EMP\_NAME VARCHAR(50),

4 DEPT\_ID INT,

5 SALARY DECIMAL(10, 2),

6 FOREIGN KEY (DEPT\_ID) REFERENCES DEPT(DEPT\_ID)

7 );

Table created.

SQL>

SQL> CREATE TABLE DEPart5 (

2 DEPT\_ID INT PRIMARY KEY,

3 DEPT\_NAME VARCHAR(50),

4 LOCATION VARCHAR(50)

5 );

Table created.

SQL> CREATE TABLE PRODUCTS (

2 PRODUCT\_ID INT PRIMARY KEY,

3 PRODUCT\_NAME VARCHAR(50),

4 PRICE DECIMAL(10, 2),

5 COLOR VARCHAR(20)

6 );

Table created.

SQL>

SQL> CREATE TABLE ORDER\_DETAILS (

2 ORDER\_ID INT PRIMARY KEY,

3 PRODUCT\_ID INT,

4 QUANTITY INT,

5 TOTAL\_PRICE DECIMAL(10, 2),

6 FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCTS(PRODUCT\_ID)

7 );

Table created.

INSERT INTO EMP51 (EMP\_ID, EMP\_NAME, DEPT\_ID, SALARY)

VALUES (&EMP\_ID, '&EMP\_NAME', &DEPT\_ID, &SALARY);

INSERT INTO DEPart5 (DEPT\_ID, DEPT\_NAME, LOCATION)

VALUES (&DEPT\_ID, '&DEPT\_NAME', '&LOCATION');

INSERT INTO PRODUCTS (PRODUCT\_ID, PRODUCT\_NAME, PRICE, COLOR)

VALUES (&PRODUCT\_ID, '&PRODUCT\_NAME', &PRICE, '&COLOR');

INSERT INTO ORDER\_DETAILS (ORDER\_ID, PRODUCT\_ID, QUANTITY, TOTAL\_PRICE)

VALUES (&ORDER\_ID, &PRODUCT\_ID, &QUANTITY, &TOTAL\_PRICE);

INSERT INTO EMP51 (EMP\_ID, EMP\_NAME, DEPT\_ID, SALARY)

VALUES (1, 'John Doe', 101, 5000.00);

INSERT INTO DEPart5 (DEPT\_ID, DEPT\_NAME, LOCATION)

VALUES (101, 'Sales', 'New York');

INSERT INTO PRODUCTS (PRODUCT\_ID, PRODUCT\_NAME, PRICE, COLOR)

VALUES (201, 'Laptop', 1200.00, 'Black');

INSERT INTO ORDER\_DETAILS (ORDER\_ID, PRODUCT\_ID, QUANTITY, TOTAL\_PRICE)

VALUES (301, 201, 2, 2400.00);

ALTER TABLE PRODUCTS ADD COLOR VARCHAR(20);

SELECT PRODUCT\_NAME || ' costs ' || PRICE || ' dollars.'

FROM PRODUCTS;

SQL> SELECT PRODUCT\_NAME || ' costs ' || PRICE || ' dollars.'

2 FROM PRODUCTS;

PRODUCT\_NAME||'COSTS'||PRICE||'DOLLARS.'

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

Laptop costs 1200 dollars.

QUESTION...

Desirable List (Practical):

. Display all the records of EMP table

Display all the records of DEPT table

Display only Name of all employees

Display Employee’s Name and salary

Display only unique departments (deptno) from EMP table

ANSWER...

SELECT \* FROM EMP51;

SELECT \* FROM DEPT;

SELECT EMP\_NAME FROM EMP;

SELECT EMP\_NAME, TOTAL FROM EMP;

SELECT DISTINCT DEPT\_ID FROM EMP;

SQL> SELECT \* FROM EMP51;

EMP\_NAME EMP\_ID PRICE1 PRICE2 TOTAL

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

tosif 101 10000 20000 30000

yasin 102 1000 2000

arman 103 10002 10003

SQL> SELECT EMP\_NAME FROM EMP;

EMP\_NAME

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

tosif

yasin

arman

SQL> SELECT \* FROM DEPart;

DEPT\_ID DEPT\_NAME LOCATION

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

1 BCA ahmedabad

2 mca surat

3 bcom Mumbai

SQL> SELECT EMP\_NAME, total FROM EMP;

EMP\_NAME TOTAL

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

tosif 30000

yasin

arman

SQL>

DAY-4

QUESTION...

Essential List (Practical)

Desirable List (Practical):

. Display all the records of EMP table

Display all the records of DEPT table

www.ljku.edu.in July-Dec2024

Display only Name of all employees

Display Employee’s Name and salary

Display only unique departments (deptno) from EMP table

ANSWER...

SELECT \* FROM EMP;

SELECT \* FROM DEPT;

SELECT EMP\_NAME FROM EMP;

SELECT EMP\_NAME, SALARY FROM EMP;

SELECT DISTINCT DEPT\_ID FROM EMP;

SQL> SELECT \* FROM EMP;

EMP\_NAME EMP\_ID PRICE1 PRICE2 TOTAL

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

tosif 101 10000 20000 30000

yasin 102 1000 2000

arman 103 10002 10003

SQL>

SQL> SELECT \* FROM DEPT;

DEPT\_ID DEPT\_NAME LOCATION

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

1 bca sanand

2 mca gomtipur

3 msc\_it jamshedpur

101 Sales New York

SQL> SELECT EMP\_NAME FROM EMP;

EMP\_NAME

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

tosif

yasin

arman

SQL> SELECT EMP\_NAME, TOTAL FROM EMP;

EMP\_NAME TOTAL

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

tosif 30000

yasin

arman

SQL>

SQL> SELECT DISTINCT DEPT\_ID FROM EMP;

DEPT\_ID

----------

1

2

3

101

DAY-5

DAY-6

DAY-7

DAY-8

DAY-9

DAY-10

DAY-11

DAY-12

QUESTION...

Essential List (Practical)

1 Use Following Tables and solve given queries:

Employee (Eid,EName,Department,Job,Salary)

Write a pl/sql block to print the name and job of an employee who is working as ‘clerk’ earning

salary of Rs 1700.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EName, Job

4 FROM Employee101

5 WHERE Job = 'clerk' AND Salary = 1700;

6

7 v\_EName VARCHAR2(50);

8 v\_Job VARCHAR2(50);

9 BEGIN

10 OPEN emp\_cursor;

11

12 LOOP

13 FETCH emp\_cursor INTO v\_EName, v\_Job;

14 EXIT WHEN emp\_cursor%NOTFOUND;

15

16 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_EName || ', Job: ' || v\_Job);

17 END LOOP;

18

19 CLOSE emp\_cursor;

20

21 EXCEPTION

22 WHEN NO\_DATA\_FOUND THEN

23 DBMS\_OUTPUT.PUT\_LINE('No employee found with the job of clerk earning Rs 1700.');

24 END;

25 /

Employee Name: John Doe, Job: clerk

Employee Name: Bob Johnson, Job: clerk

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Using above table Solve following queries.

1. Write a pl/sql block to display all employees’ name and salary whose salary is more than 2000.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EName, Salary

4 FROM Employee101

5 WHERE Salary > 2000;

6

7 v\_EName VARCHAR2(50);

8 v\_Salary DECIMAL(10, 2);

9 BEGIN

10 OPEN emp\_cursor;

11

12 LOOP

13 FETCH emp\_cursor INTO v\_EName, v\_Salary;

14 EXIT WHEN emp\_cursor%NOTFOUND;

15

16 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_EName || ', Salary: ' || v\_Salary);

17 END LOOP;

18

19 CLOSE emp\_cursor;

20

21 EXCEPTION

22 WHEN OTHERS THEN

23 DBMS\_OUTPUT.PUT\_LINE('An error occurred.');

24 END;

25 /

Employee Name: Jane Smith, Salary: 3000

Employee Name: Alice Davis, Salary: 2500

PL/SQL procedure successfully completed.

DAY-13

QUESTION 1....

Essential List (Practical)

1 Write a pl/sql block to print the total number of employees working as ‘manager’ in previously

done query.

ANSWER 1...

SQL>

DECLARE

2 v\_manager\_count NUMBER;

3 BEGIN

4 SELECT COUNT(\*)

5 INTO v\_manager\_count

6 FROM EMP2

7 WHERE job = 'MANAGER';

8

9 DBMS\_OUTPUT.PUT\_LINE('Total number of employees working as Manager: ' || v\_manager\_count);

10 END;

11 /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;

SQL> /

Total number of employees working as Manager: 0

PL/SQL procedure successfully completed.

QUESTION 1...

Desirable List (Practical):

1 Using above table Solve following queries.

1. 2. Write a pl/sql block to display all employee whose name start with ‘j’.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT ename

4 FROM EMP3

5 WHERE ename LIKE 'J%';

6 emp\_name EMP3.ename%TYPE;

7 BEGIN

8 OPEN emp\_cursor;

9 LOOP

10 FETCH emp\_cursor INTO emp\_name;

11 EXIT WHEN emp\_cursor%NOTFOUND;

12 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp\_name);

13 END LOOP;

14 CLOSE emp\_cursor;

15 END;

16 /

Employee Name: JONES

Employee Name: JAMES

Employee Name: JACKSON

PL/SQL procedure successfully completed.

DAY-14

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to print the total number of employees working as ‘manager’ in previously

done query.

ANSWER...

SQL> DECLARE

2 v\_manager\_count NUMBER;

3 BEGIN

4 SELECT COUNT(\*)

5 INTO v\_manager\_count

6 FROM EMP3

7 WHERE job = 'MANAGER';

8

9 DBMS\_OUTPUT.PUT\_LINE('Total number of employees working as Manager: ' || v\_manager\_count);

10 END;

11 /

Total number of employees working as Manager: 2

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Using above table Solve following queries.

1. Write a pl/sql block to display all employee whose name start with ‘j’.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT ename

4 FROM EMP3

5 WHERE ename LIKE 'J%';

6 emp\_name EMP3.ename%TYPE;

7 BEGIN

8 OPEN emp\_cursor;

9 LOOP

10 FETCH emp\_cursor INTO emp\_name;

11 EXIT WHEN emp\_cursor%NOTFOUND;

12 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp\_name);

13 END LOOP;

14 CLOSE emp\_cursor;

15 END;

16 /

Employee Name: JONES

Employee Name: JAMES

Employee Name: JACKSON

PL/SQL procedure successfully completed.

DAY-15

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to print total salary of employees from the employee table.

ANSWER...

SQL> DECLARE

2 v\_total\_salary NUMBER;

3 BEGIN

4 SELECT SUM(sal)

5 INTO v\_total\_salary

6 FROM EMP3;

7

8 DBMS\_OUTPUT.PUT\_LINE('Total salary of all employees: ' || v\_total\_salary);

9 END;

10 /

Total salary of all employees: 16025

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Using above table Solve following queries.

1. Write a pl/sql block to display all employees who are not in department number (deptno) 20.

ANSWER...

DECLARE

CURSOR emp\_cursor IS

SELECT ename

FROM EMP4

WHERE deptno != 20;

emp\_name EMP4.ename%TYPE; -- Change EMP to EMP4

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_name;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp\_name);

END LOOP;

CLOSE emp\_cursor;

END;

/

DAY-16

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to retrieves the details of an employees when the user is prompted to enter

employee number

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_empno NUMBER;

3 v\_ename EMP4.ename%TYPE;

4 v\_job EMP4.job%TYPE;

5 v\_mgr EMP4.mgr%TYPE;

6 v\_hiredate EMP4.hiredate%TYPE;

7 v\_sal EMP4.sal%TYPE;

8 v\_comm EMP4.comm%TYPE;

9 v\_deptno EMP4.deptno%TYPE;

10

11 BEGIN

12 v\_empno := &employee\_number;

13

14 SELECT ename, job, mgr, hiredate, sal, comm, deptno

15 INTO v\_ename, v\_job, v\_mgr, v\_hiredate, v\_sal, v\_comm, v\_deptno

16 FROM EMP4

17 WHERE empno = v\_empno;

18

19 DBMS\_OUTPUT.PUT\_LINE('Employee Number: ' || v\_empno);

20 DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_ename);

21 DBMS\_OUTPUT.PUT\_LINE('Job: ' || v\_job);

22 DBMS\_OUTPUT.PUT\_LINE('Manager: ' || v\_mgr);

23 DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || v\_hiredate);

24 DBMS\_OUTPUT.PUT\_LINE('Salary: ' || v\_sal);

25 DBMS\_OUTPUT.PUT\_LINE('Commission: ' || v\_comm);

26 DBMS\_OUTPUT.PUT\_LINE('Department Number: ' || v\_deptno);

27

28 EXCEPTION

29 WHEN NO\_DATA\_FOUND THEN

30 DBMS\_OUTPUT.PUT\_LINE('No employee found with the number: ' || v\_empno);

31 WHEN OTHERS THEN

32 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

33 END;

34 /

Enter value for employee\_number: 1

old 12: v\_empno := &employee\_number;

new 12: v\_empno := 1;

No employee found with the number: 1

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Write a PL/SQL block using CASE statement to accept the owner name from the user. The user

name can be SYS, SYSTEM, HR or SCOTT. If the owner name is SYS then print the result is ‘The

Owner is SYS’. If the owner name is SYSTEM then print the result is ‘The Owner is SYSTEM’. If the

owner name is HR then print the result is ‘The Owner is HR’. If the owner name is SCOTT then

print the result is ‘The Owners SCOTT’. Otherwise print ‘Invalid Choice

ANSWER....

PL/SQL procedure successfully completed.

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_owner\_name VARCHAR2(20);

3 v\_result VARCHAR2(50);

4 BEGIN

5 v\_owner\_name := UPPER('&owner\_name');

6

7 v\_result := CASE v\_owner\_name

8 WHEN 'SYS' THEN 'The Owner is SYS'

9 WHEN 'SYSTEM' THEN 'The Owner is SYSTEM'

10 WHEN 'HR' THEN 'The Owner is HR'

11 WHEN 'SCOTT' THEN 'The Owner is SCOTT'

12 ELSE 'Invalid Choice'

13 END;

14

15 DBMS\_OUTPUT.PUT\_LINE(v\_result);

16 END;

17 /

Enter value for owner\_name: KING

old 5: v\_owner\_name := UPPER('&owner\_name');

new 5: v\_owner\_name := UPPER('KING');

Invalid Choice

PL/SQL procedure successfully completed.

DAY-17

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to retrieves the details of an employee when the user is prompted to enter

the job\_title

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_job\_title VARCHAR2(50);

3 v\_ename EMP4.ename%TYPE;

4 v\_empno EMP4.empno%TYPE;

5 v\_mgr EMP4.mgr%TYPE;

6 v\_hiredate EMP4.hiredate%TYPE;

7 v\_sal EMP4.sal%TYPE;

8 v\_comm EMP4.comm%TYPE;

9 v\_deptno EMP4.deptno%TYPE;

10

11 BEGIN

12 v\_job\_title := '&job\_title';

13

14 SELECT empno, ename, mgr, hiredate, sal, comm, deptno

15 INTO v\_empno, v\_ename, v\_mgr, v\_hiredate, v\_sal, v\_comm, v\_deptno

16 FROM EMP4

17 WHERE job = v\_job\_title;

18

19 DBMS\_OUTPUT.PUT\_LINE('Employee Number: ' || v\_empno);

20 DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_ename);

21 DBMS\_OUTPUT.PUT\_LINE('Manager: ' || v\_mgr);

22 DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || v\_hiredate);

23 DBMS\_OUTPUT.PUT\_LINE('Salary: ' || v\_sal);

24 DBMS\_OUTPUT.PUT\_LINE('Commission: ' || v\_comm);

25 DBMS\_OUTPUT.PUT\_LINE('Department Number: ' || v\_deptno);

26

27 EXCEPTION

28 WHEN NO\_DATA\_FOUND THEN

29 DBMS\_OUTPUT.PUT\_LINE('No employee found with the job title: ' || v\_job\_title);

30 WHEN OTHERS THEN

31 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

32 END;

33 /

Enter value for job\_title: manager

old 12: v\_job\_title := '&job\_title';

new 12: v\_job\_title := 'manager';

No employee found with the job title: manager

PL/SQL procedure successfully completed.

DAY-18

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to retrieves the salary using the value the user is prompted to enter. The user

is prompted to enter job\_title. Then block test as to whether the job retrieved is an Analyst or

Manager or Clerk. For an Analyst, the salary has been increased by 5.5%, a Manager will have a

salary increased by 5%, whereas the salary of the Clerk will increase by 7.5% and the rest of the

employees will have salary increase of 8%

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_job\_title VARCHAR2(50);

3 v\_salary EMP4.sal%TYPE;

4 v\_new\_salary EMP4.sal%TYPE;

5

6 BEGIN

7 v\_job\_title := '&job\_title';

8

9 SELECT sal

10 INTO v\_salary

11 FROM EMP4

12 WHERE job = v\_job\_title;

13

14 CASE v\_job\_title

15 WHEN 'ANALYST' THEN

16 v\_new\_salary := v\_salary \* 1.055; -- Increase by 5.5%

17 WHEN 'MANAGER' THEN

18 v\_new\_salary := v\_salary \* 1.05; -- Increase by 5%

19 WHEN 'CLERK' THEN

20 v\_new\_salary := v\_salary \* 1.075; -- Increase by 7.5%

21 ELSE

22 v\_new\_salary := v\_salary \* 1.08; -- Increase by 8% for others

23 END CASE;

24

25 DBMS\_OUTPUT.PUT\_LINE('Original Salary: ' || v\_salary);

26 DBMS\_OUTPUT.PUT\_LINE('New Salary: ' || v\_new\_salary);

27

28 EXCEPTION

29 WHEN NO\_DATA\_FOUND THEN

30 DBMS\_OUTPUT.PUT\_LINE('No employee found with the job title: ' || v\_job\_title);

31 WHEN OTHERS THEN

32 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

33 END;

34 /

Enter value for job\_title: assistent\_manager

old 7: v\_job\_title := '&job\_title';

new 7: v\_job\_title := 'assistent\_manager';

No employee found with the job title: assistent\_manager

PL/SQL procedure successfully completed.

DAY-19

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to read the salary of an employee 10 and display the appropriate message if

it lies in the range of 1000 and 3000

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_empno NUMBER := 10;

3 v\_salary EMP4.sal%TYPE;

4

5 BEGIN

6 SELECT sal

7 INTO v\_salary

8 FROM EMP4

9 WHERE empno = v\_empno;

10

11 IF v\_salary BETWEEN 1000 AND 3000 THEN

12 DBMS\_OUTPUT.PUT\_LINE('The salary of employee 10 is within the range of 1000 and 3000: ' || v\_salary);

13 ELSE

14 DBMS\_OUTPUT.PUT\_LINE('The salary of employee 10 is outside the range of 1000 and 3000: ' || v\_salary);

15 END IF;

16

17 EXCEPTION

18 WHEN NO\_DATA\_FOUND THEN

19 DBMS\_OUTPUT.PUT\_LINE('No employee found with employee number: ' || v\_empno);

20 WHEN OTHERS THEN

21 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

22 END;

23 /

No employee found with employee number: 10

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Write a pl/sql block to get the details of marks (rollno, mark1, mark2, and grade) out of 100 for

marks1 and marks2 respectively. Display the grade in table marks using if statements as specified

below if stud\_percent > 70 then grade is ‘A’,if stud\_percent>60 and 70 then grade is ‘B’ else give

grade is ‘C’.

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_rollno NUMBER;

3 v\_mark1 NUMBER;

4 v\_mark2 NUMBER;

5 v\_stud\_percent NUMBER;

6 v\_grade CHAR(1);

7

8 BEGIN

9 v\_rollno := &rollno;

10 v\_mark1 := &mark1;

11 v\_mark2 := &mark2;

12

13 v\_stud\_percent := (v\_mark1 + v\_mark2) / 2;

14

15 IF v\_stud\_percent > 70 THEN

16 v\_grade := 'A';

17 ELSIF v\_stud\_percent > 60 THEN

18 v\_grade := 'B';

19 ELSE

20 v\_grade := 'C';

21 END IF;

22

23 DBMS\_OUTPUT.PUT\_LINE('Roll Number: ' || v\_rollno);

24 DBMS\_OUTPUT.PUT\_LINE('Mark 1: ' || v\_mark1);

25 DBMS\_OUTPUT.PUT\_LINE('Mark 2: ' || v\_mark2);

26 DBMS\_OUTPUT.PUT\_LINE('Student Percentage: ' || v\_stud\_percent);

27 DBMS\_OUTPUT.PUT\_LINE('Grade: ' || v\_grade);

28

29 EXCEPTION

30 WHEN OTHERS THEN

31 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

32 END;

33 /

Enter value for rollno: 1

old 9: v\_rollno := &rollno;

new 9: v\_rollno := 1;

Enter value for mark1: 2

old 10: v\_mark1 := &mark1;

new 10: v\_mark1 := 2;

Enter value for mark2: 3

old 11: v\_mark2 := &mark2;

new 11: v\_mark2 := 3;

Roll Number: 1

Mark 1: 2

Mark 2: 3

Student Percentage: 2.5

Grade: C

PL/SQL procedure successfully completed.

DAY-20

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to update the salary of the employees with 1000 when total number of

employees in a particular department is greater than 3

ANSWER...

SQL> DECLARE

2 v\_deptno NUMBER;

3 v\_emp\_count NUMBER;

4 BEGIN

5 FOR rec IN (SELECT deptno FROM EMP4 GROUP BY deptno) LOOP

6 v\_deptno := rec.deptno;

7

8 SELECT COUNT(\*) INTO v\_emp\_count

9 FROM EMP4

10 WHERE deptno = v\_deptno;

11

12 IF v\_emp\_count > 3 THEN

13 UPDATE EMP4

14 SET sal = sal + 1000

15 WHERE deptno = v\_deptno;

16 END IF;

17 END LOOP;

18

19 COMMIT;

20

21 DBMS\_OUTPUT.PUT\_LINE('Salaries updated for departments with more than 3 employees.');

22

23 EXCEPTION

24 WHEN OTHERS THEN

25 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

26 END;

27 /

Salaries updated for departments with more than 3 employees.

PL/SQL procedure successfully completed.

DAY-21

QUESTION...

Essential List (Practical)

1 Write a pl/sql block to accept job from Employee table. Give the following raise in the salary.

By 9% ,if job is clerk

By 8% ,if job is manager

By 7%, if job is Salesman.

Update the salary of the Employee table

ANSWER...

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 v\_job EMP4.job%TYPE;

3 BEGIN

4 v\_job := '&job'; -- Prompt user for job title

5

6 IF v\_job = 'CLERK' THEN

7 UPDATE EMP4

8 SET sal = sal \* 1.09

9 WHERE job = v\_job;

10 ELSIF v\_job = 'MANAGER' THEN

11 UPDATE EMP4

12 SET sal = sal \* 1.08

13 WHERE job = v\_job;

14 ELSIF v\_job = 'SALESMAN' THEN

15 UPDATE EMP4

16 SET sal = sal \* 1.07

17 WHERE job = v\_job;

18 ELSE

19 DBMS\_OUTPUT.PUT\_LINE('No salary update for the job: ' || v\_job);

20 RETURN;

21 END IF;

22

23 COMMIT;

24 DBMS\_OUTPUT.PUT\_LINE('Salaries updated for job: ' || v\_job);

25

26 EXCEPTION

27 WHEN OTHERS THEN

28 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

29 END;

30 /

Enter value for job: 20000

old 4: v\_job := '&job'; -- Prompt user for job title

new 4: v\_job := '20000'; -- Prompt user for job title

No salary update for the job: 20000

PL/SQL procedure successfully completed.

DAY-22

QUESTION...

Essential List (Practical)

1 Use following tables and write below given PL/SQL blocks.

PRODUCTS (Prod\_ID, Prod\_Name, Supplier\_ID, Cat\_ID, Unit, Price)

ORDER\_DETAILS (OrderDetail\_ID, Order\_ID, Prod\_Id, Quantity)

i. Write a PLSQL block to display total number of products ordered in Order\_ID = 3

ii. Write a PLSQL block to update the price (actual price + 5) of product with Id = 2

ANSWER...

PL/SQL procedure successfully completed.

SQL> DECLARE

2 v\_total\_quantity NUMBER;

3 BEGIN

4 SELECT SUM(Quantity)

5 INTO v\_total\_quantity

6 FROM ORDER\_DETAILS

7 WHERE Order\_ID = 3;

8

9 DBMS\_OUTPUT.PUT\_LINE('Total number of products ordered in Order\_ID = 3: ' || NVL(v\_total\_quantity, 0));

10 EXCEPTION

11 WHEN NO\_DATA\_FOUND THEN

12 DBMS\_OUTPUT.PUT\_LINE('No products found for Order\_ID = 3.');

13 WHEN OTHERS THEN

14 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

15 END;

16 /

Total number of products ordered in Order\_ID = 3: 0

PL/SQL procedure successfully completed.

SQL> BEGIN

2 UPDATE PRODUCTS2

3 SET Price = Price + 5

4 WHERE Prod\_ID = 2;

5

6 COMMIT;

7

8 IF SQL%ROWCOUNT > 0 THEN

9 DBMS\_OUTPUT.PUT\_LINE('Price updated successfully for Product ID = 2.');

10 ELSE

11 DBMS\_OUTPUT.PUT\_LINE('No product found with Product ID = 2.');

12 END IF;

13 EXCEPTION

14 WHEN OTHERS THEN

15 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

16 END;

17 /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;

SQL> /

No product found with Product ID = 2.

PL/SQL procedure successfully completed.

SQL>

DAY-23

QUESTION...

Essential List (Practical)

1 Use following tables and write below given PL/SQL blocks.

PRODUCTS (Prod\_ID, Prod\_Name, Supplier\_ID, Cat\_ID, Unit, Price)

ORDER\_DETAILS (OrderDetail\_ID, Order\_ID, Prod\_Id, Quantity)

iii. Write a PLSQL block to delete the products of Cat\_Id = 3

iv. Write a PLSQL block to insert any product whose cat\_id = 3

v. Write a PLSQL block to display Supplier\_Id and their total number of products they

supply.

ANSWER...

SQL> BEGIN

2 DELETE FROM PRODUCTS2

3 WHERE Cat\_ID = 3;

4

5 COMMIT;

6

7 DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' products deleted with Cat\_ID = 3.');

8 EXCEPTION

9 WHEN OTHERS THEN

10 ROLLBACK;

11 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

12 END;

13 /

0 products deleted with Cat\_ID = 3.

PL/SQL procedure successfully completed.

SQL> BEGIN

2 INSERT INTO PRODUCTS2 (Prod\_ID, Prod\_Name, Supplier\_ID, Cat\_ID, Unit, Price)

3 VALUES (5, 'Product E', 105, 3, 'kg', 18);

4

5 COMMIT;

6

7 DBMS\_OUTPUT.PUT\_LINE('Product inserted successfully with Cat\_ID = 3.');

8 EXCEPTION

9 WHEN OTHERS THEN

10 ROLLBACK;

11 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

12 END;

13 /

Product inserted successfully with Cat\_ID = 3.

PL/SQL procedure successfully completed.

SQL> DECLARE

2 CURSOR supplier\_cursor IS

3 SELECT Supplier\_ID, COUNT(\*) AS Total\_Products

4 FROM PRODUCTS2

5 GROUP BY Supplier\_ID;

6 v\_supplier\_id NUMBER;

7 v\_total\_products NUMBER;

8 BEGIN

9 OPEN supplier\_cursor;

10 LOOP

11 FETCH supplier\_cursor INTO v\_supplier\_id, v\_total\_products;

12 EXIT WHEN supplier\_cursor%NOTFOUND;

13 DBMS\_OUTPUT.PUT\_LINE('Supplier ID: ' || v\_supplier\_id || ', Total Products: ' || v\_total\_products);

14 END LOOP;

15 CLOSE supplier\_cursor;

16 EXCEPTION

17 WHEN OTHERS THEN

18 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

19 END;

20 /

Supplier ID: 101, Total Products: 1

Supplier ID: 102, Total Products: 1

Supplier ID: 103, Total Products: 1

Supplier ID: 104, Total Products: 1

Supplier ID: 105, Total Products: 1

PL/SQL procedure successfully completed.

DAY-24

QUESTION...

Essential List (Practical)

1 Use following tables and write below given PL/SQL blocks.

PRODUCTS (Prod\_ID, Prod\_Name, Supplier\_ID, Cat\_ID, Unit, Price)

ORDER\_DETAILS (OrderDetail\_ID, Order\_ID, Prod\_Id, Quantity)

i. Write a pl/sql block to add record twice in database using basic loop and while loop.

ANSWER...

SQL> DECLARE

2 v\_prod\_id NUMBER := 6; -- New Product ID

3 v\_prod\_name VARCHAR2(100) := 'Product F';

4 v\_supplier\_id NUMBER := 106;

5 v\_cat\_id NUMBER := 1;

6 v\_unit VARCHAR2(50) := 'kg';

7 v\_price NUMBER := 22;

8 BEGIN

9 FOR i IN 1..2 LOOP

10 INSERT INTO PRODUCTS2 (Prod\_ID, Prod\_Name, Supplier\_ID, Cat\_ID, Unit, Price)

11 VALUES (v\_prod\_id, v\_prod\_name, v\_supplier\_id, v\_cat\_id, v\_unit, v\_price);

12 END LOOP;

13

14 COMMIT;

15

16 DBMS\_OUTPUT.PUT\_LINE('Record inserted twice using FOR loop.');

17 EXCEPTION

18 WHEN OTHERS THEN

19 ROLLBACK;

20 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

21 END;

22 /

PL/SQL procedure successfully completed.

DAY-25

QUESTION...

Essential List (Practical)

1 Create a pl/sql block to create a record type as it is well-known, of all employees who earns more

than 1700 by prompting the user to enter employee no, then a block will test if the employees

earns that salary before displaying details. The record must store the employee’s

number,job,salary and annual salary

ANSWER...

SQL> DECLARE

2 TYPE emp\_record IS RECORD (

3 empno NUMBER,

4 job VARCHAR2(50),

5 sal NUMBER,

6 annual\_sal NUMBER

7 );

8 emp\_rec emp\_record;

9 v\_empno NUMBER;

10 v\_sal NUMBER;

11 BEGIN

12 v\_empno := &Enter\_Employee\_No;

13

14 SELECT SAL INTO v\_sal FROM EMP4 WHERE EMPNO = v\_empno;

15

16 IF v\_sal > 1700 THEN

17 SELECT EMPNO, JOB, SAL, SAL \* 12

18 INTO emp\_rec

19 FROM EMP4

20 WHERE EMPNO = v\_empno;

21

22 DBMS\_OUTPUT.PUT\_LINE('Employee No: ' || emp\_rec.empno);

23 DBMS\_OUTPUT.PUT\_LINE('Job: ' || emp\_rec.job);

24 DBMS\_OUTPUT.PUT\_LINE('Salary: ' || emp\_rec.sal);

25 DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || emp\_rec.annual\_sal);

26 ELSE

27 DBMS\_OUTPUT.PUT\_LINE('Employee does not earn more than 1700.');

28 END IF;

29 EXCEPTION

30 WHEN NO\_DATA\_FOUND THEN

31 DBMS\_OUTPUT.PUT\_LINE('No employee found with that number.');

32 END;

33 /

Enter value for enter\_employee\_no: 7698

old 12: v\_empno := &Enter\_Employee\_No;

new 12: v\_empno := 7698;

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical):

1 Write a pl/sql block to add record twice in database using for loop.

ANSWER...

SQL> BEGIN

2 FOR i IN 1..2 LOOP

3 BEGIN

4 INSERT INTO EMP4 (EMPNO, JOB, SAL)

5 VALUES (&Enter\_Emp\_No, '&Enter\_Job', &Enter\_Salary);

6 EXCEPTION

7 WHEN DUP\_VAL\_ON\_INDEX THEN

8 DBMS\_OUTPUT.PUT\_LINE('Duplicate entry found, skipping insert for EMPNO: ' || &Enter\_Emp\_No);

9 END;

10 END LOOP;

11 COMMIT;

12 END;

13 /

Enter value for enter\_emp\_no: 7839

Enter value for enter\_job: president

Enter value for enter\_salary: 5000

old 5: VALUES (&Enter\_Emp\_No, '&Enter\_Job', &Enter\_Salary);

new 5: VALUES (7839, 'president', 5000);

Enter value for enter\_emp\_no: 7566

old 8: DBMS\_OUTPUT.PUT\_LINE('Duplicate entry found, skipping insert for EMPNO: ' || &Enter\_Emp\_No);

new 8: DBMS\_OUTPUT.PUT\_LINE('Duplicate entry found, skipping insert for EMPNO: ' || 7566);

PL/SQL procedure successfully completed.

DAY-26

QUESTION...

Essential List (Practical)

1 Write PL/SQL block to increase the salary by 15 %( Rs. 1000) for all Employees in emp table.

(Use Cursor)

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EMPNO, SAL FROM EMP4;

4 v\_empno EMP4.EMPNO%TYPE;

5 v\_sal EMP4.SAL%TYPE;

6 BEGIN

7 OPEN emp\_cursor;

8 LOOP

9 FETCH emp\_cursor INTO v\_empno, v\_sal;

10 EXIT WHEN emp\_cursor%NOTFOUND;

11

12 UPDATE EMP4

13 SET SAL = v\_sal + (v\_sal \* 0.15) + 1000

14 WHERE EMPNO = v\_empno;

15 END LOOP;

16 CLOSE emp\_cursor;

17 COMMIT;

18 END;

19 /

PL/SQL procedure successfully completed.

DAY-27

QUESTION...

Essential List (Practical)

1 Write a PL/SQL block to increase the salary by 10% Rs. 6000) of the Employee whose

Job=’Accounting’ and display the count total number of employees whose job=’Accounting (Use

Cursor)

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EMPNO, SAL FROM EMP4 WHERE JOB = 'Accounting';

4 v\_empno EMP4.EMPNO%TYPE;

5 v\_sal EMP4.SAL%TYPE;

6 v\_count NUMBER := 0;

7 BEGIN

8 OPEN emp\_cursor;

9 LOOP

10 FETCH emp\_cursor INTO v\_empno, v\_sal;

11 EXIT WHEN emp\_cursor%NOTFOUND;

12

13 UPDATE EMP4

14 SET SAL = v\_sal + (v\_sal \* 0.10) + 6000

15 WHERE EMPNO = v\_empno;

16

17 v\_count := v\_count + 1;

18 END LOOP;

19 CLOSE emp\_cursor;

20

21 DBMS\_OUTPUT.PUT\_LINE('Total number of employees with job = ''Accounting'': ' || v\_count);

22 COMMIT;

23 END;

24 /

Total number of employees with job = 'Accounting': 0

PL/SQL procedure successfully completed.

DAY-28

QUESTION...

Essential List (Practical)

1 Write a PL/SQL block, which takes as input the department name and displays all these

employees of this department who have been working since 2010. (Use Cursor)

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EMPNO, ENAME, HIREDATE FROM EMP4

4 WHERE DEPTNO = &Enter\_Dept\_No AND EXTRACT(YEAR FROM HIREDATE) >= 2010;

5 v\_empno EMP4.EMPNO%TYPE;

6 v\_ename EMP4.ENAME%TYPE;

7 v\_hiredate EMP4.HIREDATE%TYPE;

8 BEGIN

9 OPEN emp\_cursor;

10 LOOP

11 FETCH emp\_cursor INTO v\_empno, v\_ename, v\_hiredate;

12 EXIT WHEN emp\_cursor%NOTFOUND;

13

14 DBMS\_OUTPUT.PUT\_LINE('Employee No: ' || v\_empno || ', Name: ' || v\_ename || ', Hire Date: ' || v\_hiredate);

15 END LOOP;

16 CLOSE emp\_cursor;

17 END;

18 /

Enter value for enter\_dept\_no: 30

old 4: WHERE DEPTNO = &Enter\_Dept\_No AND EXTRACT(YEAR FROM HIREDATE) >= 2010;

new 4: WHERE DEPTNO = 30 AND EXTRACT(YEAR FROM HIREDATE) >= 2010;

Employee No: 7698, Name: BLAKE, Hire Date: 01-MAY-81

Employee No: 7900, Name: JACKSON, Hire Date: 03-DEC-81

Employee No: 7654, Name: JAMES, Hire Date: 03-DEC-81

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical (Practical)

1 Write a PL/SQL Cursor to take a employee’s name as an input from the user and prepare a report

in the following format: and Give appropriate message if employee Not found.

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

EMPNO EMP\_NAME JOB SALARY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Grand Total:

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT EMPNO, ENAME, JOB, SAL FROM EMP4

4 WHERE ENAME = '&Enter\_Emp\_Name';

5

6 v\_empno EMP4.EMPNO%TYPE;

7 v\_ename EMP4.ENAME%TYPE;

8 v\_job EMP4.JOB%TYPE;

9 v\_sal EMP4.SAL%TYPE;

10 v\_total NUMBER := 0;

11 v\_found BOOLEAN := FALSE;

12 BEGIN

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

14 DBMS\_OUTPUT.PUT\_LINE('EMPNO EMP\_NAME JOB SALARY');

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

16

17 OPEN emp\_cursor;

18 LOOP

19 FETCH emp\_cursor INTO v\_empno, v\_ename, v\_job, v\_sal;

20 EXIT WHEN emp\_cursor%NOTFOUND;

21

22 v\_found := TRUE;

23 v\_total := v\_total + v\_sal;

24

25 DBMS\_OUTPUT.PUT\_LINE(v\_empno || ' ' || v\_ename || ' ' || v\_job || ' ' || v\_sal);

26 END LOOP;

27 CLOSE emp\_cursor;

28

29 IF NOT v\_found THEN

30 DBMS\_OUTPUT.PUT\_LINE('Employee not found.');

31 ELSE

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

33 DBMS\_OUTPUT.PUT\_LINE('Grand Total: ' || v\_total);

34 END IF;

35 END;

36 /

Enter value for enter\_emp\_name: clark

old 4: WHERE ENAME = '&Enter\_Emp\_Name';

new 4: WHERE ENAME = 'clark';

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

EMPNO EMP\_NAME JOB SALARY

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

Employee not found.

PL/SQL procedure successfully completed.

DAY-29

QUESTION...

Essential List (Practical)

1 Create table:

EmployeeInformation: (Emp\_No. , Emp\_Name , Emp\_Dept. , Emp\_Salary)

In employee information table update the employee name 'AADITYA' department 'Program

Developer' changes to 'Web Developer' using implicit cursor.

ANSWER...

SQL> CREATE TABLE EmployeeInformation (

2 Emp\_No NUMBER PRIMARY KEY,

3 Emp\_Name VARCHAR2(50),

4 Emp\_Dept VARCHAR2(50),

5 Emp\_Salary NUMBER

6 );

Table created.

SQL>

SQL> BEGIN

2 UPDATE EmployeeInformation

3 SET Emp\_Dept = 'Web Developer'

4 WHERE Emp\_Name = 'AADITYA' AND Emp\_Dept = 'Program Developer';

5

6 COMMIT;

7 END;

8 /

PL/SQL procedure successfully completed.

DAY-30

QUESTION...

Essential List (Practical)

1 Create table:

EmployeeInformation: (Emp\_No. , Emp\_Name , Emp\_Dept. , Emp\_Salary)

In employee information table write PLSQL block to retrieve the details of those employee whose

salary is more than 20000 using explicit cursor.

ANSWER...

SQL>

DECLARE

2 CURSOR emp\_cursor IS

3 SELECT Emp\_No, Emp\_Name, Emp\_Dept, Emp\_Salary

4 FROM EmployeeInformation1

5 WHERE Emp\_Salary > 20000;

6

7 v\_Emp\_No EmployeeInformation.Emp\_No%TYPE;

8 v\_Emp\_Name EmployeeInformation.Emp\_Name%TYPE;

9 v\_Emp\_Dept EmployeeInformation.Emp\_Dept%TYPE;

10 v\_Emp\_Salary EmployeeInformation.Emp\_Salary%TYPE;

11 BEGIN

12 OPEN emp\_cursor;

13 LOOP

14 FETCH emp\_cursor INTO v\_Emp\_No, v\_Emp\_Name, v\_Emp\_Dept, v\_Emp\_Salary;

15 EXIT WHEN emp\_cursor%NOTFOUND;

16 DBMS\_OUTPUT.PUT\_LINE('Emp\_No: ' || v\_Emp\_No || ', Emp\_Name: ' || v\_Emp\_Name || ', Emp\_Dept: ' || v\_Emp\_Dept || ', Emp\_Salary: ' || v\_Emp\_Salary);

17 END LOOP;

18 CLOSE emp\_cursor;

19 END;

20 /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;

SQL> /

Emp\_No: 101, Emp\_Name: John Doe, Emp\_Dept: HR, Emp\_Salary: 25000

Emp\_No: 103, Emp\_Name: Emily Davis, Emp\_Dept: IT, Emp\_Salary: 30000

Emp\_No: 104, Emp\_Name: Michael Brown, Emp\_Dept: Marketing, Emp\_Salary: 22000

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical (Practical)

1 Using explicit cursor retrieves the records of first 5 employees one by one from employee

information table.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT Emp\_No, Emp\_Name, Emp\_Dept, Emp\_Salary

4 FROM EmployeeInformation1

5 WHERE ROWNUM <= 5;

6

7 v\_Emp\_No EmployeeInformation1.Emp\_No%TYPE;

8 v\_Emp\_Name EmployeeInformation1.Emp\_Name%TYPE;

9 v\_Emp\_Dept EmployeeInformation1.Emp\_Dept%TYPE;

10 v\_Emp\_Salary EmployeeInformation1.Emp\_Salary%TYPE;

11 BEGIN

12 OPEN emp\_cursor;

13 LOOP

14 FETCH emp\_cursor INTO v\_Emp\_No, v\_Emp\_Name, v\_Emp\_Dept, v\_Emp\_Salary;

15 EXIT WHEN emp\_cursor%NOTFOUND;

16 DBMS\_OUTPUT.PUT\_LINE('Emp\_No: ' || v\_Emp\_No || ', Emp\_Name: ' || v\_Emp\_Name || ', Emp\_Dept: ' || v\_Emp\_Dept || ', Emp\_Salary: ' || v\_Emp\_Salary);

17 END LOOP;

18 CLOSE emp\_cursor;

19 END;

20 /

Emp\_No: 101, Emp\_Name: John Doe, Emp\_Dept: HR, Emp\_Salary: 25000

Emp\_No: 102, Emp\_Name: Jane Smith, Emp\_Dept: Finance, Emp\_Salary: 18000

Emp\_No: 103, Emp\_Name: Emily Davis, Emp\_Dept: IT, Emp\_Salary: 30000

Emp\_No: 104, Emp\_Name: Michael Brown, Emp\_Dept: Marketing, Emp\_Salary: 22000

Emp\_No: 105, Emp\_Name: Sarah Wilson, Emp\_Dept: IT, Emp\_Salary: 19000

PL/SQL procedure successfully completed.

DAY-31

QUESTION...

Essential List (Practical)

1 Create table:

EmployeeInformation: (Emp\_No. , Emp\_Name , Emp\_Dept. , Emp\_Salary)

Display employee number wise first two employee details using for loop cursor.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor IS

3 SELECT Emp\_No, Emp\_Name, Emp\_Dept, Emp\_Salary

4 FROM EmployeeInformation1

5 ORDER BY Emp\_No;

6

7 emp\_count NUMBER := 0;

8 BEGIN

9 FOR emp\_record IN emp\_cursor LOOP

10 emp\_count := emp\_count + 1;

11 DBMS\_OUTPUT.PUT\_LINE('Emp\_No: ' || emp\_record.Emp\_No ||

12 ', Emp\_Name: ' || emp\_record.Emp\_Name ||

13 ', Emp\_Dept: ' || emp\_record.Emp\_Dept ||

14 ', Emp\_Salary: ' || emp\_record.Emp\_Salary);

15

16 IF emp\_count = 2 THEN

17 EXIT;

18 END IF;

19 END LOOP;

20 END;

21 /

Emp\_No: 101, Emp\_Name: John Doe, Emp\_Dept: HR, Emp\_Salary: 25000

Emp\_No: 102, Emp\_Name: Jane Smith, Emp\_Dept: Finance, Emp\_Salary: 18000

PL/SQL procedure successfully completed.

DAY-32

QUESTION...

Essential List (Practical)

1 Create table:

EmployeeInformation: (Emp\_No. , Emp\_Name , Emp\_Dept. , Emp\_Salary)

Display employee information from emp\_information table whose emp\_no four (4) using

parameterized cursor.

ANSWER...

SQL> DECLARE

2 CURSOR emp\_cursor(p\_Emp\_No EmployeeInformation1.Emp\_No%TYPE) IS

3 SELECT Emp\_No, Emp\_Name, Emp\_Dept, Emp\_Salary

4 FROM EmployeeInformation1

5 WHERE Emp\_No = p\_Emp\_No;

6

7 v\_Emp\_No EmployeeInformation1.Emp\_No%TYPE;

8 v\_Emp\_Name EmployeeInformation1.Emp\_Name%TYPE;

9 v\_Emp\_Dept EmployeeInformation1.Emp\_Dept%TYPE;

10 v\_Emp\_Salary EmployeeInformation1.Emp\_Salary%TYPE;

11 BEGIN

12 OPEN emp\_cursor(4);

13 FETCH emp\_cursor INTO v\_Emp\_No, v\_Emp\_Name, v\_Emp\_Dept, v\_Emp\_Salary;

14 IF emp\_cursor%FOUND THEN

15 DBMS\_OUTPUT.PUT\_LINE('Emp\_No: ' || v\_Emp\_No ||

16 ', Emp\_Name: ' || v\_Emp\_Name ||

17 ', Emp\_Dept: ' || v\_Emp\_Dept ||

18 ', Emp\_Salary: ' || v\_Emp\_Salary);

19 ELSE

20 DBMS\_OUTPUT.PUT\_LINE('No employee found with Emp\_No 4.');

21 END IF;

22 CLOSE emp\_cursor;

23 END;

24 /

No employee found with Emp\_No 4.

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Using Explicit Cursor to display the top five highest paid workers who are specialized in

‘POLISHING’.

Wroker(WrokerId,Name,Wage\_Per\_Hour,Specialized\_In,)

ANSWER...

SQL> DECLARE

2 CURSOR worker\_cursor IS

3 SELECT WorkerId, Name, Wage\_Per\_Hour, Specialized\_In

4 FROM Worker

5 WHERE Specialized\_In = 'POLISHING'

6 ORDER BY Wage\_Per\_Hour DESC

7 FETCH FIRST 5 ROWS ONLY;

8

9 v\_WorkerId Worker.WorkerId%TYPE;

10 v\_Name Worker.Name%TYPE;

11 v\_Wage\_Per\_Hour Worker.Wage\_Per\_Hour%TYPE;

12 v\_Specialized\_In Worker.Specialized\_In%TYPE;

13 BEGIN

14 OPEN worker\_cursor;

15 LOOP

16 FETCH worker\_cursor INTO v\_WorkerId, v\_Name, v\_Wage\_Per\_Hour, v\_Specialized\_In;

17 EXIT WHEN worker\_cursor%NOTFOUND;

18 DBMS\_OUTPUT.PUT\_LINE('WorkerId: ' || v\_WorkerId ||

19 ', Name: ' || v\_Name ||

20 ', Wage\_Per\_Hour: ' || v\_Wage\_Per\_Hour ||

21 ', Specialized\_In: ' || v\_Specialized\_In);

22 END LOOP;

23 CLOSE worker\_cursor;

24 END;

25 /

WorkerId: 4, Name: Charlie Davis, Wage\_Per\_Hour: 35, Specialized\_In: POLISHING

WorkerId: 2, Name: Alice Johnson, Wage\_Per\_Hour: 30, Specialized\_In: POLISHING

WorkerId: 5, Name: Emily Wilson, Wage\_Per\_Hour: 28, Specialized\_In: POLISHING

WorkerId: 7, Name: Sarah White, Wage\_Per\_Hour: 27, Specialized\_In: POLISHING

WorkerId: 1, Name: John Smith, Wage\_Per\_Hour: 25, Specialized\_In: POLISHING

PL/SQL procedure successfully completed.

DAY-33

QUESTION...

Essential List (Practical)

1 Write PL/SQL block to increase the salary by 15 %( Rs. 1000) for all Employees in emp table.

Raise a user defined exception if an employee is not found.

ANSWER...

SQL> DECLARE

2 no\_employees\_found EXCEPTION;

3 v\_row\_count NUMBER;

4

5 BEGIN

6 UPDATE emp4

7 SET SAL = SAL +

8 GREATEST(0.15 \* SAL, 1000);

9

10 v\_row\_count := SQL%ROWCOUNT;

11

12 IF v\_row\_count = 0 THEN

13 RAISE no\_employees\_found;

14 END IF;

15

16 DBMS\_OUTPUT.PUT\_LINE(v\_row\_count || ' employee(s) salary updated.');

17

18 EXCEPTION

19 WHEN no\_employees\_found THEN

20 DBMS\_OUTPUT.PUT\_LINE('No employees found in the emp4 table.');

21 END;

22 /

7 employee(s) salary updated.

PL/SQL procedure successfully completed.

DAY-34

QUESTION...

Essential List (Practical)

1 Write a PL/SQL block to increase the salary by 10 %( Rs. 6000) of the Employee whose

Job=’Accounting’ and display the count total number of employees whose job=’Accounting.

Raise and exception for Job title

ANSWER...

DECLARE

2 no\_employees\_found EXCEPTION;

3 v\_count NUMBER;

4

5 BEGIN

6 SELECT COUNT(\*)

7 INTO v\_count

8 FROM Employee

9 WHERE JOB = 'ACCOUNTING';

10

11 IF v\_count = 0 THEN

12 RAISE no\_employees\_found;

13 END IF;

14

15 UPDATE Employee

16 SET SAL = SAL + GREATEST(0.10 \* SAL, 6000)

17 WHERE JOB = 'ACCOUNTING';

18

19 DBMS\_OUTPUT.PUT\_LINE(v\_count || ' employee(s) found in Accounting. Salary updated.');

20

21 EXCEPTION

22 WHEN no\_employees\_found THEN

23 DBMS\_OUTPUT.PUT\_LINE('No employees found with the job title Accounting.');

24 END;

25 /

2 employee(s) found in Accounting. Salary updated.

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Write a user defined exception for program where if student’s marks are less than 0 then

appropriate error message must be shown as exception.

Write a PLSQL block to find the largest of three numbers

ANSWER...

SQL> DECLARE

2 invalid\_marks EXCEPTION;

3 v\_marks1 NUMBER;

4 v\_marks2 NUMBER;

5 v\_marks3 NUMBER;

6 v\_largest NUMBER;

7

8 BEGIN

9 SELECT Marks INTO v\_marks1 FROM student1 WHERE StudentID = 1; -- Replace with actual ID for the first student

10 SELECT Marks INTO v\_marks2 FROM student1 WHERE StudentID = 2; -- Replace with actual ID for the second student

11 SELECT Marks INTO v\_marks3 FROM student1 WHERE StudentID = 3; -- Replace with actual ID for the third student

12

13 IF v\_marks1 < 0 THEN

14 RAISE invalid\_marks;

15 END IF;

16

17 IF v\_marks2 < 0 THEN

18 RAISE invalid\_marks;

19 END IF;

20

21 IF v\_marks3 < 0 THEN

22 RAISE invalid\_marks;

23 END IF;

24

25 v\_largest := GREATEST(v\_marks1, v\_marks2, v\_marks3);

26

27 DBMS\_OUTPUT.PUT\_LINE('The largest mark is: ' || v\_largest);

28

29 EXCEPTION

30 WHEN invalid\_marks THEN

31 DBMS\_OUTPUT.PUT\_LINE('Error: Marks cannot be less than 0.');

32 END;

33 /

The largest mark is: 89

PL/SQL procedure successfully completed.

DAY-35

QUESTION...

Essential List (Practical)

1 Create a procedure to display the employee name whose employeeno is accepted by the user.

ANSWER...

PL/SQL procedure successfully completed.

SQL> CREATE OR REPLACE PROCEDURE DisplayEmployeeName(p\_empno IN NUMBER) IS

2 v\_emp\_name VARCHAR2(50);

3 BEGIN

4 SELECT ENAME INTO v\_emp\_name

5 FROM emp4

6 WHERE EMPNO = p\_empno;

7

8 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_emp\_name);

9 EXCEPTION

10 WHEN NO\_DATA\_FOUND THEN

11 DBMS\_OUTPUT.PUT\_LINE('No employee found with Employee Number: ' || p\_empno);

12 END DisplayEmployeeName;

13 /

Procedure created.

DAY-36

QUESTION...

Essential List (Practical)

1 Use Following Tables:

Emp\_Master (Emp\_Code, Emp\_Name, Birth\_Date)

Dept\_Master (Dept\_Code, Dept\_Name, Budget)

Salary (Dept\_Code, Emp\_Code, Salary)

Write a pl/sql block to call a procedure that counts total employees in each department and

display them. Also handles the exception where no record found.

ANSWER...

SQL> CREATE OR REPLACE PROCEDURE CountEmployeesInDepartment IS

2 CURSOR emp\_cursor IS

3 SELECT D.Dept\_Name, COUNT(E.Emp\_Code) AS Employee\_Count

4 FROM Dept\_Master D

5 LEFT JOIN Salary S ON D.Dept\_Code = S.Dept\_Code

6 LEFT JOIN Emp\_Master E ON S.Emp\_Code = E.Emp\_Code

7 GROUP BY D.Dept\_Name;

8

9 v\_dept\_name VARCHAR2(50);

10 v\_emp\_count NUMBER;

11 v\_found\_records BOOLEAN := FALSE;

12 BEGIN

13 OPEN emp\_cursor;

14 LOOP

15 FETCH emp\_cursor INTO v\_dept\_name, v\_emp\_count;

16 EXIT WHEN emp\_cursor%NOTFOUND;

17 DBMS\_OUTPUT.PUT\_LINE('Department: ' || v\_dept\_name || ', Total Employees: ' || v\_emp\_count);

18 v\_found\_records := TRUE;

19 END LOOP;

20 CLOSE emp\_cursor;

21

22 IF NOT v\_found\_records THEN

23 RAISE NO\_DATA\_FOUND;

24 END IF;

25

26 EXCEPTION

27 WHEN NO\_DATA\_FOUND THEN

28 DBMS\_OUTPUT.PUT\_LINE('No records found for any department.');

29 END CountEmployeesInDepartment;

30 /

Procedure created.

SQL> BEGIN

2 CountEmployeesInDepartment;

3 END;

4 /

Department: HR, Total Employees: 1

Department: IT, Total Employees: 2

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Write a stored procedure that uses an INOUT parameter and an IN parameter. The user will

supply 'M' or 'F' through IN parameter (emp\_gender) to count a number of male or female

from Employee table. The INOUT parameter (mfgender) will return the result to a user.

ANSWER...

SQL> CREATE OR REPLACE PROCEDURE CountGenderEmployees (

2 emp\_gender IN CHAR,

3 mfgender IN OUT NUMBER

4 ) AS

5 BEGIN

6 SELECT COUNT(\*)

7 INTO mfgender

8 FROM EMPLOYEE55

9 WHERE E = emp\_gender;

10

11 DBMS\_OUTPUT.PUT\_LINE('Count of ' || CASE WHEN emp\_gender = 'M' THEN 'Male' ELSE 'Female' END || ' Employees: ' || mfgender);

12 EXCEPTION

13 WHEN NO\_DATA\_FOUND THEN

14 mfgender := 0; -- Set the count to 0 if no data found

15 DBMS\_OUTPUT.PUT\_LINE('Count of ' || CASE WHEN emp\_gender = 'M' THEN 'Male' ELSE 'Female' END || ' Employees: ' || mfgender);

16 END CountGenderEmployees;

17 /

Procedure created.

DAY-37

QUESTION...

Essential List (Practical)

1 Create a procedure to display the sum of salary of the employees whose job is accepted by the

user.

ANSWER...

SQL> CREATE OR REPLACE PROCEDURE SumSalaryByJob (

2 p\_job IN VARCHAR2

3 ) AS

4 total\_salary NUMBER; -- Variable to hold the sum of salaries

5 BEGIN

6 SELECT SUM(SAL)

7 INTO total\_salary

8 FROM EMP4

9 WHERE JOB = p\_job;

10

11 DBMS\_OUTPUT.PUT\_LINE('Total salary for job "' || p\_job || '": ' || NVL(total\_salary, 0));

12 EXCEPTION

13 WHEN NO\_DATA\_FOUND THEN

14 DBMS\_OUTPUT.PUT\_LINE('No employees found for job: ' || p\_job);

15 WHEN OTHERS THEN

16 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

17 END SumSalaryByJob;

18 /

Procedure created.

SQL> BEGIN

2 SumSalaryByJob('MANAGER'); -- Replace 'MANAGER' with the desired job title

3 END;

4 /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;

SQL> /

Total salary for job "MANAGER": 23494.24

PL/SQL procedure successfully completed.

DAY-38

QUESTION...

Essential List (Practical)

1 Create a procedure which gets the name of the employee when the employee id is passed

using IN and OUT parameter.

ANSWER...

SQL> CREATE OR REPLACE PROCEDURE GetEmployeeName (

2 p\_emp\_id IN NUMBER,

3 p\_emp\_name OUT VARCHAR2

4 ) AS

5 BEGIN

6 SELECT ENAME

7 INTO p\_emp\_name

8 FROM EMP4

9 WHERE EMPNO = p\_emp\_id;

10

11 IF p\_emp\_name IS NULL THEN

12 p\_emp\_name := 'Employee not found';

13 END IF;

14 EXCEPTION

15 WHEN NO\_DATA\_FOUND THEN

16 p\_emp\_name := 'Employee not found';

17 WHEN OTHERS THEN

18 p\_emp\_name := 'An error occurred: ' || SQLERRM;

19 END GetEmployeeName;

20 /

Procedure created.

SQL> DECLARE

2 emp\_name VARCHAR2(50); -- Variable to hold the employee name

3 BEGIN

4 GetEmployeeName(7839, emp\_name); -- Replace 7839 with the desired employee ID

5 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp\_name);

6 END;

7 /

Employee Name: KING

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Create a procedure to that takes as input the department number and passes two output

parameters-one giving the total number of employees in that department and the other giving

the maximum salary in that department

ANSWER...

SQL> CREATE OR REPLACE PROCEDURE GetDepartmentStats (

2 p\_dept\_no IN NUMBER,

3 p\_total\_employees OUT NUMBER,

4 p\_max\_salary OUT NUMBER

5 ) AS

6 BEGIN

7 SELECT COUNT(\*), MAX(SAL)

8 INTO p\_total\_employees, p\_max\_salary

9 FROM emp4

10 WHERE DEPTNO = p\_dept\_no;

11

12 IF p\_total\_employees = 0 THEN

13 p\_max\_salary := NULL; -- No employees in the department

14 END IF;

15

16 EXCEPTION

17 WHEN NO\_DATA\_FOUND THEN

18 p\_total\_employees := 0;

19 p\_max\_salary := NULL;

20 WHEN OTHERS THEN

21 p\_total\_employees := 0;

22 p\_max\_salary := NULL;

23 END GetDepartmentStats;

24 /

Procedure created.

SQL> DECLARE

2 total\_employees NUMBER;

3 max\_salary NUMBER;

4 BEGIN

5 GetDepartmentStats(10, total\_employees, max\_salary); -- Replace 10 with the desired department number

6 DBMS\_OUTPUT.PUT\_LINE('Total Employees in Department 10: ' || total\_employees);

7 DBMS\_OUTPUT.PUT\_LINE('Maximum Salary in Department 10: ' || NVL(max\_salary, 0));

8 END;

9 /

Total Employees in Department 10: 2

Maximum Salary in Department 10: 11588.41

PL/SQL procedure successfully completed.

DAY-39

QUESTION..

Essential List (Practical)

1 Create a function to return employee name whose salary is above 50000 from the

EmployeeInformation table.

ANSWER...

SQL> CREATE OR REPLACE FUNCTION GetEmployeeNameAboveSalary (

2 min\_salary IN NUMBER

3 ) RETURN VARCHAR2 AS

4 emp\_name VARCHAR2(100);

5 BEGIN

6 SELECT Emp\_Name

7 INTO emp\_name

8 FROM EmployeeInformation1

9 WHERE Emp\_Salary > min\_salary

10 AND ROWNUM = 1; -- Return the first employee found

11

12 RETURN emp\_name; -- Return the employee name

13 EXCEPTION

14 WHEN NO\_DATA\_FOUND THEN

15 RETURN 'No employee found';s

16 WHEN OTHERS THEN

17 RETURN 'An error occurred: ' || SQLERRM;

18 END GetEmployeeNameAboveSalary;

19 /

Function created.

SQL> DECLARE

2 employee\_name VARCHAR2(100);

3 BEGIN

4 employee\_name := GetEmployeeNameAboveSalary(50000);

employee with salary above 50,000

5 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || employee\_name);

6 END;

7 /

Employee Name: No employee found

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Create a function which returns total number of female employees from employee table

ANSWER...

SQL> CREATE OR REPLACE FUNCTION CountFemaleEmployees

2 RETURN NUMBER AS

3 female\_count NUMBER;

4 BEGIN

5 SELECT COUNT(\*)

6 INTO female\_count

7 FROM emp66

8 WHERE GENDER = 'F';

9

10 RETURN female\_count;

11 EXCEPTION

12 WHEN OTHERS THEN

13 RETURN 0;

14 END CountFemaleEmployees;

15 /

Function created.

SQL>

SQL> DECLARE

2 total\_females NUMBER;

3 BEGIN

4 total\_females := CountFemaleEmployees();

5 DBMS\_OUTPUT.PUT\_LINE('Total Female Employees: ' || total\_females);

6 END;

7 /

Total Female Employees: 2

PL/SQL procedure successfully completed.

DAY-40

QUESTION...

Essential List (Practical)

1 Use Following Tables:

Emp\_Master (Emp\_Code, Emp\_Name, Birth\_Date)

Dept\_Master (Dept\_Code, Dept\_Name, Budget)

Salary (Dept\_Code, Emp\_Code, Salary)

Write a pl/sql block to call a function that returns total employees in each department and

display them. Also handles the exception where no record found

ANSWER...

SQL> CREATE OR REPLACE FUNCTION CountEmployeesByDept(dept\_code IN NUMBER)

2 RETURN NUMBER AS

3 emp\_count NUMBER;

4 BEGIN

5 SELECT COUNT(\*)

6 INTO emp\_count

7 FROM Salary

8 WHERE Dept\_Code = dept\_code;

9

10 RETURN emp\_count;

11 EXCEPTION

12 WHEN NO\_DATA\_FOUND THEN

13 RETURN 0;

14 END CountEmployeesByDept;

15 /

Function created.

SQL>

SQL> DECLARE

2 dept\_rec Dept\_Master%ROWTYPE;

3 total\_employees NUMBER;

4 BEGIN

5 FOR dept\_rec IN (SELECT \* FROM Dept\_Master) LOOP

6 total\_employees := CountEmployeesByDept(dept\_rec.Dept\_Code);

7 DBMS\_OUTPUT.PUT\_LINE('Department: ' || dept\_rec.Dept\_Name || ', Total Employees: ' || total\_employees);

8 END LOOP;

9 EXCEPTION

10 WHEN OTHERS THEN

11 DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

12 END;

13 /

Department: HR, Total Employees: 3

Department: IT, Total Employees: 3

PL/SQL procedure successfully completed.

QUESTION....

Desirable List (Practical)

1 Use following tables and do as directed:

Movie (movie\_id, movie name, date\_of\_release)

www.ljku.edu.in July-Dec2024

Screen (screen\_id, location, max\_capacity)

Current (movie\_id,screen\_id, date\_of\_arrival, date\_of\_closure)

1. Consider the above table and write a function to return the movie name which arrived

today. .

ANSWER...

SQL> CREATE OR REPLACE FUNCTION GetMoviesArrivedToday

2 RETURN VARCHAR2 AS

3 movie\_names VARCHAR2(4000);

4 BEGIN

5 SELECT LISTAGG(m.movie\_name, ', ') WITHIN GROUP (ORDER BY m.movie\_name)

6 INTO movie\_names

7 FROM Movie m

8 JOIN Current5 c ON m.movie\_id = c.movie\_id

9 WHERE c.date\_of\_arrival = TRUNC(SYSDATE);

10

11 IF movie\_names IS NULL THEN

12 RETURN 'No movies arrived today.';

13 ELSE

14 RETURN movie\_names;

15 END IF;

16 EXCEPTION

17 WHEN OTHERS THEN

18 RETURN 'An error occurred: ' || SQLERRM;

19 END GetMoviesArrivedToday;

20 /

Function created.

SQL> DECLARE

2 movies\_today VARCHAR2(4000);

3 BEGIN

4 movies\_today := GetMoviesArrivedToday();

5 DBMS\_OUTPUT.PUT\_LINE('Movies that arrived today: ' || movies\_today);

6 END;

7 /

Movies that arrived today: Avatar, Interstellar

PL/SQL procedure successfully completed.

DAY-41

QUESTION...

Essential List (Practical/Theory)

1 Create a package which stores the above functions and procedures.

2 Use following tables and do as directed:

Movie (movie\_id, movie name, date\_of\_release)

Screen (screen\_id, location, max\_capacity)

Current (movie\_id,screen\_id, date\_of\_arrival, date\_of\_closure)

. Write a function that will return max\_capacity of a screen by providing Screen\_Id as a

parameter

ANSWER...

SQL> CREATE OR REPLACE PACKAGE movie\_screen\_pkg AS

2 FUNCTION get\_max\_capacity(p\_screen\_id IN NUMBER) RETURN NUMBER;

3 END movie\_screen\_pkg;

4 /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY movie\_screen\_pkg AS

2 FUNCTION get\_max\_capacity(p\_screen\_id IN NUMBER) RETURN NUMBER IS

3 v\_max\_capacity NUMBER;

4 BEGIN

5 SELECT max\_capacity INTO v\_max\_capacity

6 FROM Screen

7 WHERE screen\_id = p\_screen\_id;

8

9 RETURN v\_max\_capacity;

10 EXCEPTION

11 WHEN NO\_DATA\_FOUND THEN

12 RETURN NULL; -- or raise an error if preferred

13 END get\_max\_capacity;

14 END movie\_screen\_pkg;

15 /

Package body created.

SQL> DECLARE

2 v\_capacity NUMBER;

3 BEGIN

4 v\_capacity := movie\_screen\_pkg.get\_max\_capacity(1); -- Replace 1 with the desired screen\_id

5 DBMS\_OUTPUT.PUT\_LINE('Max Capacity: ' || v\_capacity);

6 END;

7 /

Max Capacity: 200

PL/SQL procedure successfully completed.

QUESTION...

Desirable List (Practical)

1 Use following tables and do as directed:

Movie (movie\_id, movie\_name, date\_of\_release)

Create a trigger on the movie table where trigger should be fired when the new movie is

added..

ANSWER...

SQL> SHOW ERRORS TRIGGER trg\_after\_insert\_movie;

Errors for TRIGGER TRG\_AFTER\_INSERT\_MOVIE:

LINE/COL ERROR

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

3/5 PL/SQL: SQL Statement ignored

3/17 PL/SQL: ORA-00942: table or view does not exist

SQL> CREATE TABLE Movie\_Log (

2 log\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

3 movie\_id NUMBER,

4 movie\_name VARCHAR2(100),

5 date\_of\_release DATE,

6 action\_date DATE

7 );

Table created.

SQL> CREATE OR REPLACE TRIGGER trg\_after\_insert\_movie

2 AFTER INSERT ON Movie

3 FOR EACH ROW

4 BEGIN

5 INSERT INTO Movie\_Log (movie\_id, movie\_name, date\_of\_release, action\_date)

6 VALUES (:NEW.movie\_id, :NEW.movie\_name, :NEW.date\_of\_release, SYSDATE);

7 END trg\_after\_insert\_movie;

8 /

Trigger created.

DAY-42

QUESTION...

Essential List (Practical/Theory)

1 What is trigger? Explain its types.

2 Create a trigger on the movie table where trigger should be fired when the movie is deleted

ANSWER...

SQL> SHOW ERRORS TRIGGER trg\_after\_delete\_movie;

Errors for TRIGGER TRG\_AFTER\_DELETE\_MOVIE:

LINE/COL ERROR

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

3/5 PL/SQL: SQL Statement ignored

3/17 PL/SQL: ORA-00942: table or view does not exist

SQL> CREATE TABLE Movie\_Deletion\_Log (

2 log\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

3 movie\_id NUMBER,

4 movie\_name VARCHAR2(100),

5 date\_of\_deletion DATE

6 );

Table created.

SQL> CREATE OR REPLACE TRIGGER trg\_after\_delete\_movie

2 AFTER DELETE ON Movie

3 FOR EACH ROW

4 BEGIN

5 INSERT INTO Movie\_Deletion\_Log (movie\_id, movie\_name, date\_of\_deletion)

6 VALUES (:OLD.movie\_id, :OLD.movie\_name, SYSDATE);

7 END;

8 /

Trigger created.

DAY-43

QUESTION...

Essential List (Practical/Theory)

1 Use following tables and do as directed:

College(collegeId,collegeName,location)

Course(CourseId, CourseName,CourseDuration,EligibleStudents)

Class(collegeId,CourseId,ClassStartDate,ClassEndDate)

. Write a function that will return Eligible Students of a course by providing CourseId as a

parameter

ANSWER...

SQL> CREATE OR REPLACE FUNCTION get\_eligible\_students(p\_course\_id IN NUMBER)

2 RETURN NUMBER

3 IS

4 v\_eligible\_students NUMBER;

5 BEGIN

6 SELECT EligibleStudents

7 INTO v\_eligible\_students

8 FROM Course

9 WHERE CourseId = p\_course\_id;

10

11 RETURN v\_eligible\_students;

12 EXCEPTION

13 WHEN NO\_DATA\_FOUND THEN

14 RETURN NULL; -- or raise an error if preferred

15 END get\_eligible\_students;

16 /

Function created.

SQL> DECLARE

2 v\_students NUMBER;

3 BEGIN

4 v\_students := get\_eligible\_students(1); -- Replace 1 with the desired CourseId

5 DBMS\_OUTPUT.PUT\_LINE('Eligible Students: ' || v\_students);

6 END;

7 /

Eligible Students: 120

PL/SQL procedure successfully completed.

DAY-44

QUESTION...

Essential List (Practical/Theory)

1 Use following tables and do as directed:

College(collegeId,collegeName,location)

Course (CourseId, CourseName, CourseDuration,

EligibleStudents,CourseStartDt.,CourseEndDt.)

Class(collegeId,CourseId,ClassStartDate,ClassEndDate)

Consider the above table and write a function to return the courseName which started from

today today

ANSWER...

SQL> SHOW ERRORS FUNCTION get\_courses\_started\_today;

Errors for FUNCTION GET\_COURSES\_STARTED\_TODAY:

LINE/COL ERROR

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

7/5 PL/SQL: SQL Statement ignored

9/17 PL/SQL: ORA-00904: "COURSESTARTDT": invalid identifier

SQL> CREATE OR REPLACE FUNCTION get\_courses\_started\_today

2 RETURN SYS\_REFCURSOR

3 IS

4 v\_courses SYS\_REFCURSOR;

5 BEGIN

6 OPEN v\_courses FOR

7 SELECT CourseName

8 FROM Course

9 WHERE TRUNC(CourseStartDt) = TRUNC(SYSDATE);

10

11 RETURN v\_courses;

12 END;

13 /

Function created.

DAY-45

QUESTION...

Essential List (Practical/Theory)

1 Subject\_Master (Sub\_code, Sub\_name)

Student\_Master (Roll\_no, Stud\_Name, Gender, DOB, Address)

Result (Roll\_No, Sub\_code, Marks)

Find out the average score in percentage for each subject.

Find out the students whose birthday falls into leap year.

ANSWER...

SQL> SELECT

2 sm.Sub\_code,

3 sm.Sub\_name,

4 AVG(r.Marks) AS Average\_Marks,

5 (AVG(r.Marks) / 100) \* 100 AS Average\_Percentage

6 FROM

7 Subject\_Master sm

8 JOIN

9 Result r ON sm.Sub\_code = r.Sub\_code

10 GROUP BY

11 sm.Sub\_code, sm.Sub\_name;

SUB\_CODE SUB\_NAME AVERAGE\_MARKS AVERAGE\_PERCENTAGE

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

101 Mathematics 84.3333333 84.3333333

102 Physics 87.3333333 87.3333333

103 Chemistry

86.5 86.5

SQL> SELECT

2 Roll\_no,

3 Stud\_Name,

4 Gender,

5 DOB,

6 Address

7 FROM

8 Student\_Master

9 WHERE

10 MOD(EXTRACT(YEAR FROM DOB), 4) = 0

11 AND (MOD(EXTRACT(YEAR FROM DOB), 100) != 0 OR MOD(EXTRACT(YEAR FROM DOB), 400) = 0);

ROLL\_NO STUD\_NAME GENDER DOB

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

ADDRESS

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

1 John Doe Male 29-FEB-00

123 Main St, New York

3 Emily Davis Female 01-MAR-04

789 Oak St, Chicago

4 Michael Brown Male 28-FEB-00

321 Pine St, Houston

ROLL\_NO STUD\_NAME GENDER DOB

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

ADDRESS

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

5 Sarah Wilson Female 29-FEB-00

654 Maple St, Phoenix