**2**

---Ques#1

create a procedure called add\_job to insert a new job into

the jobs table.provide the id and job title using tow parameters.

A.

CREATE OR REPLACE PROCEDURE add\_job

(p\_id VARCHAR2,

p\_title VARCHAR2)

IS

BEGIN

INSERT INTO jobs (job\_id, job\_title)

VALUES (p\_id, p\_title);

END add\_job;

/

--invoke

EXECUTE add\_job('IT\_DBA', 'Database Administrator');

--

SELECT \* FROM jobs WHERE job\_id='IT\_DBA';

---Ques#2

create a procedure called upd\_jobs to update the job title.

provide the job id and a new title usng to parameters . including the

A.

CREATE OR REPLACE PROCEDURE upd\_job

(p\_id VARCHAR2,

p\_title VARCHAR2)

IS

BEGIN

update jobs

set job\_title = p\_title

where job\_id = p\_id;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202, 'No Job Updated');

END IF;

END upd\_job;

/

--invoke

EXECUTE upd\_job('IT\_DBA', 'Data Administrator');

--

SELECT \*FROM jobs WHERE job\_id='IT\_DBA';

---Ques#3

create a procedure called DEL\_JOB TO delete a job include the

necessary exception handing code if no job is delete

A.

CREATE OR REPLACE PROCEDURE del\_job

(p\_jobid VARCHAR2)

IS

BEGIN

DELETE FROM jobs

WHERE job\_id = p\_jobid;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20203, 'No Jobs Delete');

END IF;

END del\_job;

/

--invoke

EXECUTE del\_job('IT\_DBA');

select \* from jobs where job\_id ='IT\_DBA';

---Ques#4

Create a procedure that returns a value from the salary and job\_id columns

for a specfied employee id remove syntax error, if any and then recoplic ta code

A.

CREATE OR REPLACE PROCEDURE get\_employee

(p\_emp\_id IN NUMBER,

p\_sal OUT NUMBER,

p\_job\_id OUT VARCHAR2)

IS

BEGIN

SELECT salary, job\_id

INTO p\_sal, p\_job\_id FROM employees

WHERE employee\_id=p\_emp\_id;

END ;

/

B.

DECLARE

v\_sal NUMBER;

v\_jid VARCHAR2(50);

BEGIN

get\_employee(120, v\_sal, v\_jid);

DBMS\_OUTPUT.PUT\_LINE(v\_sal||' '||v\_jid);

END;

/

**3**

---Ques#1

create and compile a function called get\_job to return a job title

A.

CREATE OR REPLACE FUNCTION get\_job

(p\_jobid IN VARCHAR2)

RETURN VARCHAR2

IS

v\_title VARCHAR2(100);

BEGIN

SELECT job\_title INTO v\_title FROM jobs

WHERE job\_id=p\_jobid;

RETURN v\_title;

END get\_job;

/

B.

VARIABLE b\_title VARCHAR2(35)

EXECUTE :b\_title:=get\_job('SA\_REP');

PRINT b\_title

---Ques#2

create the get\_annual\_comp function. which accepts parameter values for the

monthly salary and commission, dither or both values passed val

A.

CREATE OR REPLACE FUNCTION get\_annual\_comp

(p\_sal IN NUMBER,

p\_comm IN NUMBER)

RETURN NUMBER

IS

BEGIN

RETURN (NVL(p\_sal,0)\*12+(NVL(p\_comm,0) \* nvl (p\_sal,0)\*12));

END get\_annual\_comp;

/

B.

SELECT employee\_id, last\_name,

get\_annual\_comp(salary, commission\_pct) "Annual Compensation"

FROM employees

WHERE department\_id=30;

---Ques#3

create a function call valid\_deptid to validate a specificted department id and

return a boolean value of true if the department exit

A.

CREATE OR REPLACE FUNCTION valid\_deptid

(p\_deptid IN NUMBER)

RETURN BOOLEAN IS

v\_num NUMBER;

BEGIN

SELECT 1 INTO v\_num FROM departments

WHERE department\_id = p\_deptid;

RETURN TRUE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END valid\_deptid;

/

C.

EXECUTE add\_employee('Jane', 'Harris', 'JHARRIS', 'ST\_MAN', NULL, NULL,NULL, p\_dept\_id=>15);

D.

EXECUTE add\_employee('Jane', 'Harris', 'JHARRIS', 'ST\_MAN', NULL, NULL,NULL, p\_dept\_id= 40);

Question 1.

1. Create a procedure called ADD\_DEPT to insert a new department into the departments table.

Provide the DEPARTMENT NAME and LOCATION ID of the department using two parameters.

-- Create Procedure

CREATE OR REPLACE PROCEDURE ADD\_DEPT

(p\_dept IN VARCHAR2,

p\_loc IN NUMBER) IS

BEGIN

INSERT INTO departments

VALUES(DEPARTMENTS\_SEQ.NEXTVAL, p\_dept, null, p\_loc);

END;

/

-- Invoke the Procedure

EXECUTE add\_dept('Test 4', 3000);

-- Check data

SELECT \* FROM departments

WHERE department\_name = 'Test 4';

---delete

delete from departments where department\_id =520;

Question 2.

2. Write a function that returns the total number of EMPLOYEES in the employee table.

-- Create Function

CREATE OR REPLACE FUNCTION emp\_total

RETURN NUMBER

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count

FROM employees;

RETURN v\_count;

END;

/

-- Invoke the Procedure

execute DBMS\_OUTPUT.PUT\_LINE('Total employees: '||emp\_total);

Question 3.

3. Create a procedure called GET\_EMPLOYEE to query the EMPLOYEES table,

retrieving the salary and job ID for an employee when provided with the employee ID.

-- Create Procedure

CREATE OR REPLACE PROCEDURE GET\_EMPLOYEE

(p\_id IN NUMBER) IS

v\_sal NUMBER;

v\_job VARCHAR2(50);

BEGIN

SELECT salary, job\_id INTO v\_sal, v\_job

FROM employees

WHERE employee\_id = p\_id;

DBMS\_OUTPUT.PUT\_LINE('Salary is: '||v\_sal||' '||'Job is: '||v\_job);

END;

/

-- Invoke Procedure

EXECUTE GET\_EMPLOYEE(105);

Question 4.

4.Write a trigger to prevent employees from being deleted during business hours (weekdays

from 9:00 AM to 6:00 PM).

-- Create Trigger

CREATE OR REPLACE TRIGGER emp\_del\_trig

BEFORE DELETE ON employees

DECLARE

v\_day VARCHAR2(3) := TO\_CHAR(SYSDATE, 'DY');

v\_time NUMBER := TO\_CHAR(SYSDATE, 'HH24');

BEGIN

IF (v\_day NOT IN ('FRI', 'SAT')) AND (v\_time BETWEEN 9 AND 18) THEN

RAISE\_APPLICATION\_ERROR (-20100, 'Employee records cannot be deleted

on weekdays during business hours');

END IF;

END;

/

-- Delete

DELETE FROM employees

WHERE employee\_id = 104;

Question 5.

5.Write a trigger to generate a auto increment number for the department id before insert on the departments table.

-- Create Trigger

CREATE OR REPLACE TRIGGER trg\_dept\_id

BEFORE INSERT ON departments

FOR EACH ROW

BEGIN

IF (:new.department\_id IS null) THEN

:new.department\_id := DEPARTMENTS\_SEQ.NEXTVAL;

END IF;

END;

/

-- Insert Data

INSERT INTO departments (department\_name, location\_id)

VALUES('Test22', 3000);

-- Check

SELECT \* FROM departments

WHERE department\_name = 'Test22';