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;
/
```

---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';
```