**Question 1: Create a Procedure to Insert Employee Data**

CREATE OR REPLACE PROCEDURE insert\_employee (

p\_emp\_id NUMBER,

p\_emp\_name VARCHAR2,

p\_department VARCHAR2,

p\_salary NUMBER

) AS

BEGIN

INSERT INTO EMPLOYEES (EMP\_ID, EMP\_NAME, DEPARTMENT, SALARY)

VALUES (p\_emp\_id, p\_emp\_name, p\_department, p\_salary);

END;

/

**Question 2: Create a Procedure to Update Employee Salary**

CREATE OR REPLACE PROCEDURE update\_salary (

p\_emp\_id NUMBER

) AS

v\_salary EMPLOYEES.SALARY%TYPE;

BEGIN

SELECT SALARY INTO v\_salary FROM EMPLOYEES WHERE EMP\_ID = p\_emp\_id;

IF v\_salary < 5000 THEN

v\_salary := v\_salary \* 1.10;

ELSIF v\_salary BETWEEN 5000 AND 10000 THEN

v\_salary := v\_salary \* 1.075;

ELSE

v\_salary := v\_salary \* 1.05;

END IF;

UPDATE EMPLOYEES

SET SALARY = v\_salary

WHERE EMP\_ID = p\_emp\_id;

END;

/

**Question 3: Use a Cursor to Display Employee Names**

DECLARE

CURSOR emp\_cursor IS

SELECT EMP\_NAME FROM EMPLOYEES;

v\_emp\_name EMPLOYEES.EMP\_NAME%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_emp\_name;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_emp\_name);

END LOOP;

CLOSE emp\_cursor;

END;

/

**Question 4: Create a View for Employees with High Salary**

CREATE OR REPLACE VIEW high\_salary\_employees AS

SELECT \*

FROM EMPLOYEES

WHERE SALARY > 10000;

Question 5: Create a Function to Calculate Bonus

Ans:

CREATE OR REPLACE FUNCTION calculate\_bonus (

p\_salary NUMBER

) RETURN NUMBER IS

v\_bonus NUMBER;

BEGIN

IF p\_salary < 5000 THEN

v\_bonus := p\_salary \* 0.10;

ELSIF p\_salary BETWEEN 5000 AND 10000 THEN

v\_bonus := p\_salary \* 0.075;

ELSE

v\_bonus := p\_salary \* 0.05;

END IF;

RETURN v\_bonus;

END;

/

**Question 6: Create a Trigger to Log Employee Insertions**

CREATE OR REPLACE TRIGGER log\_employee\_insert

AFTER INSERT ON EMPLOYEES

FOR EACH ROW

BEGIN

INSERT INTO EMPLOYEE\_LOG (LOG\_ID, EMP\_ID, LOG\_DATE)

VALUES (LOG\_SEQ.NEXTVAL, :NEW.EMP\_ID, SYSDATE);

END;

/

**Question 7: Orders and Order\_Items Tables**

A) Create a view that returns the sales revenues by customers. The values of the credit column are

5% of the total sales revenues.

Ans:

CREATE OR REPLACE VIEW sales\_revenues\_by\_customers AS

SELECT

c.customer\_id,

c.customer\_name,

SUM(oi.quantity \* oi.unit\_price) AS total\_sales,

SUM(oi.quantity \* oi.unit\_price) \* 0.05 AS credit

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

JOIN

order\_items oi ON o.order\_id = oi.order\_id

GROUP BY

c.customer\_id, c.customer\_name;

B) Write the PL/ANS: query to develop an anonymous block

DECLARE

v\_budget NUMBER := 1000000;

CURSOR cust\_cursor IS

SELECT customer\_id FROM sales\_revenues\_by\_customers ORDER BY total\_sales DESC;

v\_customer\_id sales\_revenues\_by\_customers.customer\_id%TYPE;

BEGIN

-- Reset credit limits

UPDATE customers SET credit\_limit = 0;

OPEN cust\_cursor;

LOOP

FETCH cust\_cursor INTO v\_customer\_id;

EXIT WHEN cust\_cursor%NOTFOUND;

-- Update new credit limit

UPDATE customers

SET credit\_limit = credit\_limit + (v\_budget / (SELECT COUNT(\*) FROM

sales\_revenues\_by\_customers))

WHERE customer\_id = v\_customer\_id;

v\_budget := v\_budget - (v\_budget / (SELECT COUNT(\*) FROM sales\_revenues\_by\_customers));

END LOOP;

CLOSE cust\_cursor;

END;

/

**Question 8: Show the Uses of Implicit Cursor**

DECLARE

v\_count INTEGER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM employees;

DBMS\_OUTPUT.PUT\_LINE('Total number of employees: ' || v\_count);

END;

/

**Question 9: Create a Cursor to Display Name and Salary**

DECLARE

CURSOR emp\_cursor (p\_salary NUMBER) IS

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary < p\_salary;

v\_first\_name employees.first\_name%TYPE;

v\_last\_name employees.last\_name%TYPE;

v\_salary employees.salary%TYPE;

BEGIN

OPEN emp\_cursor(10000);

LOOP

FETCH emp\_cursor INTO v\_first\_name, v\_last\_name, v\_salary;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_first\_name || ' ' || v\_last\_name || ': ' || v\_salary);

END LOOP;

CLOSE emp\_cursor;

END;

/

**Question 10: Create a Trigger to Check for Duplicate Values**

CREATE OR REPLACE TRIGGER check\_duplicate\_emp\_id

BEFORE INSERT OR UPDATE ON employees

FOR EACH ROW

DECLARE

v\_count INTEGER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM employees

WHERE employee\_id = :NEW.employee\_id;

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Duplicate employee\_id found.');

END IF;

END;

/

**Question 11: Procedure for Selecting Records with Filters**

CREATE OR REPLACE PROCEDURE select\_employees\_by\_salary (

p\_salary NUMBER

) AS

BEGIN

FOR emp IN (SELECT \* FROM ib\_employee WHERE salary = p\_salary) LOOP

DBMS\_OUTPUT.PUT\_LINE(emp.first\_name || ' ' || emp.last\_name || ': ' || emp.salary);

END LOOP;

END;

/

**Question 12: Increment Employee's Salary**

BEGIN

UPDATE EMPLOYEES

SET SALARY = SALARY + 1000

WHERE EMPLOYEE\_ID = 102;

END;