**PL/SQL ASSIGNMENT**

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

**Write a PL/SQL procedure named insert\_employee to insert employee data into the**

**EMPLOYEES table:**

** Table structure: EMPLOYEES (EMP\_ID NUMBER, EMP\_NAME VARCHAR2(100),**

**DEPARTMENT VARCHAR2(50), SALARY NUMBER)**

create table employees (

emp\_id number constraint employees\_pk primary key,

emp\_name varchar2(255),

dept varchar2(255),

salary number

)

CREATE PROCEDURE insert\_employees (

p\_emp\_id IN NUMBER,

p\_emp\_name IN VARCHAR2,

p\_department IN VARCHAR2,

p\_salary IN NUMBER

) AS

BEGIN

INSERT INTO EMPLOYEES (emp\_id, emp\_name, dept,salary)

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

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20001, 'An error occurred while inserting the employee data: ' || SQLERRM);

END insert\_employees;

BEGIN

insert\_employee(1, 'Nithialakshmi', 'HR', 50000);

END;

**2. Create a Procedure to Update Employee Salary Write a PL/SQL procedure named update\_salary to update an employee's salary based on their current salary: • If the current salary is less than 5000, increase it by 10%. • If the current salary is between 5000 and 10000, increase it by 7.5%. • If the current salary is more than 10000, increase it by 5%.**

CREATE PROCEDURE update\_salary (

p\_emp\_id IN NUMBER

) AS

v\_current\_salary EMPLOYEES.SALARY%TYPE;

v\_new\_salary EMPLOYEES.SALARY%TYPE;

BEGIN

SELECT SALARY INTO v\_current\_salary

FROM EMPLOYEES

WHERE EMP\_ID = p\_emp\_id;

IF v\_current\_salary < 5000 THEN

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

ELSIF v\_current\_salary BETWEEN 5000 AND 10000 THEN

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

ELSE

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

END IF;

UPDATE EMPLOYEES

SET SALARY = v\_new\_salary

WHERE EMP\_ID = p\_emp\_id;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found');

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20003, 'An error occurred while updating the salary: ' || SQLERRM);

END update\_salary;

/

SELECT SALARY INTO v\_current\_salary

FROM EMPLOYEES

WHERE EMP\_ID = p\_emp\_id;

CREATE PROCEDURE update\_salary (

p\_emp\_id IN NUMBER

) AS

v\_current\_salary EMPLOYEES.SALARY%TYPE;

v\_new\_salary EMPLOYEES.SALARY%TYPE;

BEGIN

SELECT SALARY INTO v\_current\_salary

FROM EMPLOYEES

WHERE EMP\_ID = p\_emp\_id

FOR UPDATE;

IF v\_current\_salary < 5000 THEN

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

ELSIF v\_current\_salary BETWEEN 5000 AND 10000 THEN

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

ELSE

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

END IF;

UPDATE EMPLOYEES

SET SALARY = v\_new\_salary

WHERE EMP\_ID = p\_emp\_id;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found');

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20003, 'An error occurred while updating the salary: ' || SQLERRM);

END update\_salary;

/

BEGIN

update\_salary(1);

END;

/

select \* from employees;

**3. Use a Cursor to Display Employee Names**

**Write a PL/SQL block using a cursor to fetch and display all employee names from the EMPLOYEES table.**

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;

EXCEPTION

WHEN OTHERS THEN

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

END;

/

**4. Create a View for Employees with High Salary**

**Write a SQL statement to create a view named high\_salary\_employees that displays employees earning more than 10000.**

CREATE VIEW high\_salary\_employees AS

SELECT EMP\_ID, EMP\_NAME, DEPT, SALARY

FROM EMPLOYEES

WHERE SALARY > 10000;

SELECT \* FROM high\_salary\_employees;

**5. Create a Function to Calculate Bonus**

**Write a PL/SQL function named calculate\_bonus to calculate the bonus based on an**

**employee's salary:**

** Employees earning less than 5000 get a bonus of 10% of their salary.**

** Employees earning between 5000 and 10000 get a bonus of 7.5% of their salary.**

** Employees earning more than 10000 get a bonus of 5% of their salary.**

**CREATE FUNCTION calculate\_bonus (**

**p\_salary IN 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;**

**EXCEPTION**

**WHEN OTHERS THEN**

**RETURN NULL;**

**END calculate\_bonus;**

**/**

**SELECT calculate\_bonus(4500) FROM DUAL;**

**DECLARE**

**v\_salary NUMBER := 7500;**

**v\_bonus NUMBER;**

**BEGIN**

**v\_bonus := calculate\_bonus(v\_salary);**

**DBMS\_OUTPUT.PUT\_LINE('The bonus is: ' || v\_bonus);**

**END;**

**/**

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

**Write a PL/SQL trigger named log\_employee\_insert to log whenever an employee is inserted into the EMPLOYEES table.**

CREATE TABLE EMPLOYEE\_LOG (

LOG\_ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

EMP\_ID NUMBER,

EMP\_NAME VARCHAR2(100),

DEPARTMENT VARCHAR2(50),

SALARY NUMBER,

INSERT\_DATE DATE

);

CREATE TRIGGER log\_employee\_insert

AFTER INSERT ON EMPLOYEES

FOR EACH ROW

BEGIN

INSERT INTO EMPLOYEE\_LOG (EMP\_ID, EMP\_NAME, DEPT, SALARY, INSERT\_DATE)

VALUES (:NEW.EMP\_ID, :NEW.EMP\_NAME, :NEW.DEPT, :NEW.SALARY, SYSDATE);

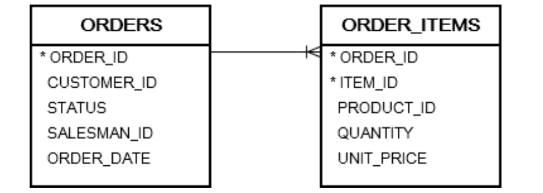
END;

/

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

VALUES (11, 'Jayashree’, 'HR', 6000);

**Question 7:Consider the orders and order\_items tables from the sample database.**



CREATE TABLE ORDERS (

ORDER\_ID NUMBER PRIMARY KEY,

CUSTOMER\_ID NUMBER,

STATUS VARCHAR2(20),

SALESMAN\_ID NUMBER,

ORDER\_DATE DATE

);

CREATE TABLE ORDER\_ITEMS (

ORDER\_ID NUMBER,

ITEM\_ID NUMBER,

PRODUCT\_ID NUMBER,

QUANTITY NUMBER,

UNIT\_PRICE NUMBER,

PRIMARY KEY (ORDER\_ID, ITEM\_ID),

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS (ORDER\_ID)

);

**A)Create a view that returns the sales revenues by customers. The values of the credit column are 5% of the total sales revenues.**

CREATE VIEW sales\_revenues\_by\_customers AS

SELECT

o.CUSTOMER\_ID,

SUM(oi.QUANTITY \* oi.UNIT\_PRICE) AS total\_sales\_revenue,

SUM(oi.QUANTITY \* oi.UNIT\_PRICE) \* 0.05 AS credit

FROM

ORDERS o

JOIN ORDER\_ITEMS oi ON o.ORDER\_ID = oi.ORDER\_ID

GROUP BY

o.CUSTOMER\_ID;

SELECT \* FROM sales\_revenues\_by\_customers;

**B) Write the PL/SQL query to develop an anonymous block which:**

**1. Reset the credit limits of all customers to zero**.

UPDATE Orders SET credit = 0;

**2. Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.**

DECLARE

CURSOR customer\_cursor IS

SELECT CUSTOMER\_ID, Total\_Sales\_Revenue

FROM Sales\_Revenue\_By\_Customers

ORDER BY Total\_Sales\_Revenue DESC;

customer\_rec customer\_cursor%ROWTYPE;

budget NUMBER := 1000000;

remaining\_budget NUMBER := 1000000;

BEGIN

UPDATE CUSTOMERS

SET CREDIT\_LIMIT = 0;

OPEN customer\_cursor;

LOOP

FETCH customer\_cursor INTO customer\_rec;

EXIT WHEN customer\_cursor%NOTFOUND;

IF remaining\_budget >= customer\_rec.Total\_Sales\_Revenue \* 0.05 THEN

UPDATE CUSTOMERS

SET CREDIT\_LIMIT = customer\_rec.Total\_Sales\_Revenue \* 0.05

WHERE CUSTOMER\_ID = customer\_rec.CUSTOMER\_ID;

remaining\_budget := remaining\_budget - (customer\_rec.Total\_Sales\_Revenue \* 0.05);

ELSE

UPDATE CUSTOMERS

SET CREDIT\_LIMIT = remaining\_budget

WHERE CUSTOMER\_ID = customer\_rec.CUSTOMER\_ID;

remaining\_budget := 0;

EXIT;

END IF;

END LOOP;

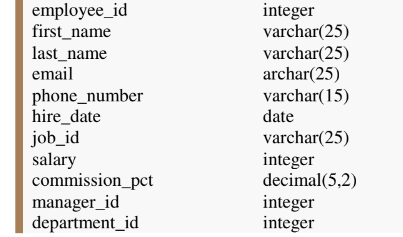
CLOSE customer\_cursor;

END;

/

**Question 8:Write a program in PL/SQL to show the uses of implicit cursor without using any attribute.**

**Table: employees**



CREATE TABLE EMPLOYEES (

EMPLOYEE\_ID INTEGER PRIMARY KEY,

FIRST\_NAME VARCHAR2(25),

LAST\_NAME VARCHAR2(25),

EMAIL VARCHAR2(25),

PHONE\_NUMBER VARCHAR2(15),

HIRE\_DATE DATE,

JOB\_ID VARCHAR2(25),

SALARY INTEGER,

COMMISSION\_PCT NUMBER(5,2),

MANAGER\_ID INTEGER,

DEPARTMENT\_ID INTEGER

);

DECLARE

-- Local variables to hold employee details

v\_employee\_id EMPLOYEES.EMPLOYEE\_ID%TYPE;

v\_first\_name EMPLOYEES.FIRST\_NAME%TYPE;

v\_last\_name EMPLOYEES.LAST\_NAME%TYPE;

v\_email EMPLOYEES.EMAIL%TYPE;

v\_phone\_number EMPLOYEES.PHONE\_NUMBER%TYPE;

v\_hire\_date EMPLOYEES.HIRE\_DATE%TYPE;

v\_job\_id EMPLOYEES.JOB\_ID%TYPE;

v\_salary EMPLOYEES.SALARY%TYPE;

v\_commission\_pct EMPLOYEES.COMMISSION\_PCT%TYPE;

v\_manager\_id EMPLOYEES.MANAGER\_ID%TYPE;

v\_department\_id EMPLOYEES.DEPARTMENT\_ID%TYPE;

-- Cursor variable to hold the cursor

CURSOR emp\_cursor IS

SELECT \* FROM EMPLOYEES;

BEGIN

-- Open the cursor

OPEN emp\_cursor;

-- Loop through each row in the cursor

LOOP

FETCH emp\_cursor INTO

v\_employee\_id,

v\_first\_name,

v\_last\_name,

v\_email,

v\_phone\_number,

v\_hire\_date,

v\_job\_id,

v\_salary,

v\_commission\_pct,

v\_manager\_id,

v\_department\_id;

EXIT WHEN emp\_cursor%NOTFOUND;

-- Print employee details

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_employee\_id);

DBMS\_OUTPUT.PUT\_LINE('First Name: ' || v\_first\_name);

DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || v\_last\_name);

DBMS\_OUTPUT.PUT\_LINE('Email: ' || v\_email);

DBMS\_OUTPUT.PUT\_LINE('Phone Number: ' || v\_phone\_number);

DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || v\_hire\_date);

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

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

DBMS\_OUTPUT.PUT\_LINE('Commission Pct: ' || v\_commission\_pct);

DBMS\_OUTPUT.PUT\_LINE('Manager ID: ' || v\_manager\_id);

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || v\_department\_id);

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

END LOOP;

-- Close the cursor

CLOSE emp\_cursor;

EXCEPTION

WHEN OTHERS THEN

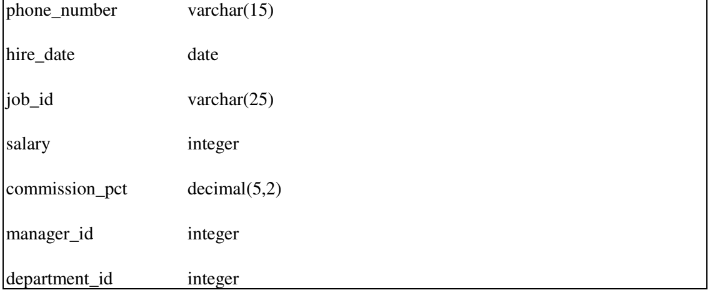
DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

**Question 9:Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passed-in parameter value.**

**Table: employees**

CREATE TABLE EMPLOYEES (

EMPLOYEE\_ID INTEGER PRIMARY KEY,

FIRST\_NAME VARCHAR2(25),

LAST\_NAME VARCHAR2(25),

EMAIL VARCHAR2(25),

PHONE\_NUMBER VARCHAR2(15),

HIRE\_DATE DATE,

JOB\_ID VARCHAR2(25),

SALARY INTEGER,

COMMISSION\_PCT NUMBER(5,2),

MANAGER\_ID INTEGER,

DEPARTMENT\_ID INTEGER

);

DECLARE

p\_salary\_limit NUMBER := 50000; -- Replace with desired value or pass as a parameter

CURSOR emp\_cursor IS

SELECT FIRST\_NAME, SALARY

FROM EMPLOYEES

WHERE SALARY < p\_salary\_limit;

emp\_record emp\_cursor%ROWTYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_record;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('First Name: ' || emp\_record.FIRST\_NAME);

DBMS\_OUTPUT.PUT\_LINE('Salary: ' || emp\_record.SALARY);

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

END LOOP;

CLOSE emp\_cursor;

END;

/

**Question 10:Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.**

CREATE TRIGGER check\_duplicate\_email

BEFORE INSERT OR UPDATE ON EMPLOYEES

FOR EACH ROW

DECLARE

v\_count INTEGER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM EMPLOYEES

WHERE EMAIL = :NEW.EMAIL

AND EMPLOYEE\_ID <> :NEW.EMPLOYEE\_ID;

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Duplicate email address detected: ' || :NEW.EMAIL);

END IF;

END;

/

**Question 11:Write a PL/SQL procedure for selecting some records from the database**

**using some parameters as filters.**

** Consider that we are fetching details of employees from ib\_employee table where salary is a parameter for filter.**

CREATE TABLE IB\_EMPLOYEE (

EMPLOYEE\_ID INTEGER PRIMARY KEY,

FIRST\_NAME VARCHAR2(25),

LAST\_NAME VARCHAR2(25),

EMAIL VARCHAR2(25) UNIQUE,

PHONE\_NUMBER VARCHAR2(15),

HIRE\_DATE DATE,

JOB\_ID VARCHAR2(25),

SALARY INTEGER,

COMMISSION\_PCT NUMBER(5,2),

MANAGER\_ID INTEGER,

DEPARTMENT\_ID INTEGER

);

INSERT INTO IB\_EMPLOYEE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID)

VALUES (1, 'Jayashree', 'Narayanan', 'jayashree@gmail.com', '555-1234', TO\_DATE('2020-01-15', 'YYYY-MM-DD'), 'DEV', 50000, 0.10, NULL, 10);

INSERT INTO IB\_EMPLOYEE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID)

VALUES (2, 'Kamal', 'Krishna', 'kamal@gmail.com', '555-5678', TO\_DATE('2019-03-22', 'YYYY-MM-DD'), 'HR', 60000, 0.05, 1, 20);

INSERT INTO IB\_EMPLOYEE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID)

VALUES (3, 'Arun', 'Kumar', 'arunkumar@gmail.com', '555-8765', TO\_DATE('2021-07-30', 'YYYY-MM-DD'), 'FIN', 50000, 0.07, 1, 30);

select \* from IB\_EMPLOYEE;

CREATE PROCEDURE fetch\_employees\_by\_salary(p\_salary IN NUMBER) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Fetching employees with salary: ' || p\_salary);

FOR emp\_rec IN (

SELECT \*

FROM IB\_EMPLOYEE

WHERE SALARY = p\_salary

) LOOP

-- Display employee details

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp\_rec.EMPLOYEE\_ID);

DBMS\_OUTPUT.PUT\_LINE('First Name: ' || emp\_rec.FIRST\_NAME);

DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || emp\_rec.LAST\_NAME);

DBMS\_OUTPUT.PUT\_LINE('Email: ' || emp\_rec.EMAIL);

DBMS\_OUTPUT.PUT\_LINE('Phone Number: ' || emp\_rec.PHONE\_NUMBER);

DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || emp\_rec.HIRE\_DATE);

DBMS\_OUTPUT.PUT\_LINE('Job ID: ' || emp\_rec.JOB\_ID);

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

DBMS\_OUTPUT.PUT\_LINE('Commission Pct: ' || emp\_rec.COMMISSION\_PCT);

DBMS\_OUTPUT.PUT\_LINE('Manager ID: ' || emp\_rec.MANAGER\_ID);

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || emp\_rec.DEPARTMENT\_ID);

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

END LOOP;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No employees found with the specified salary.');

END IF;

END;

/

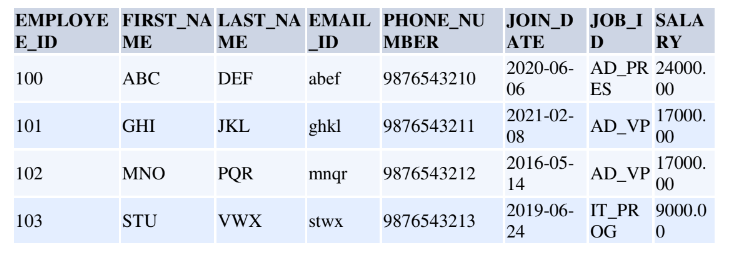
BEGIN

fetch\_employees\_by\_salary(50000);

END;

/

**Question 12:Write PL/SQL code block to increment the employee’s salary by 1000 whose employee\_id is 102 from the given table below.**



CREATE TABLE EMPLOYE (

EMPLOYEE\_ID INTEGER PRIMARY KEY,

FIRST\_NAME VARCHAR2(25),

LAST\_NAME VARCHAR2(25),

EMAIL VARCHAR2(25),

PHONE\_NUMBER VARCHAR2(15),

JOIN\_DATE DATE,

JOB\_ID VARCHAR2(25),

SALARY NUMBER

);

INSERT INTO EMPLOYE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, JOIN\_DATE, JOB\_ID, SALARY)

VALUES (100, 'ABC', 'DEF', 'abef', '9876543210', TO\_DATE('2020-06-06', 'YYYY-MM-DD'), 'AD\_PR', 24000.00);

INSERT INTO EMPLOYE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, JOIN\_DATE, JOB\_ID, SALARY)

VALUES (101, 'GHI', 'JKL', 'ghkl', '9876543211', TO\_DATE('2021-02-08', 'YYYY-MM-DD'), 'AD\_VP', 17000.00);

INSERT INTO EMPLOYE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, JOIN\_DATE, JOB\_ID, SALARY)

VALUES (102, 'MNO', 'PQR', 'mnqr', '9876543212', TO\_DATE('2016-05-14', 'YYYY-MM-DD'), 'AD\_VP', 17000.00);

INSERT INTO EMPLOYE (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, JOIN\_DATE, JOB\_ID, SALARY)

VALUES (103, 'STU', 'VWX', 'stwx', '9876543213', TO\_DATE('2019-06-24', 'YYYY-MM-DD'), 'IT\_PROG', 9000.00);

BEGIN

UPDATE EMPLOYE

SET SALARY = SALARY + 1000

WHERE EMPLOYEE\_ID = 102;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully for employee ID 102.');

EXCEPTION

WHEN OTHERS THEN

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

END;

/