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(01, ‘Hemamalani’, 'Development', 28000);
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(-02, 'Employee ID not found');
   WHEN OTHERS THEN
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-03, '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;
/
```

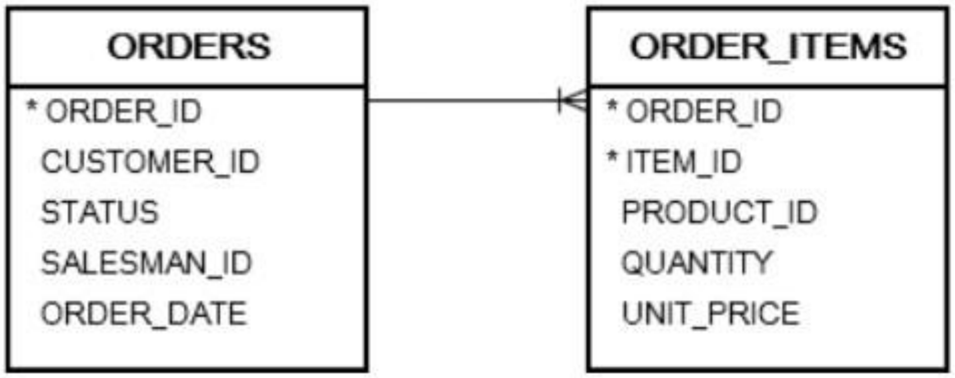
Question 6: Create a Trigger to Log Employee Insertions

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 (07, ‘Hari’, 'IT', 22000);
```

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;
```

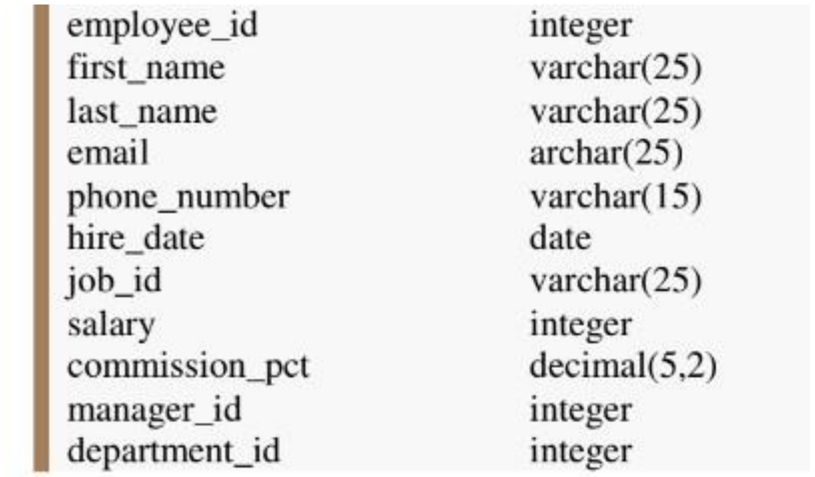
2. Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.

```
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

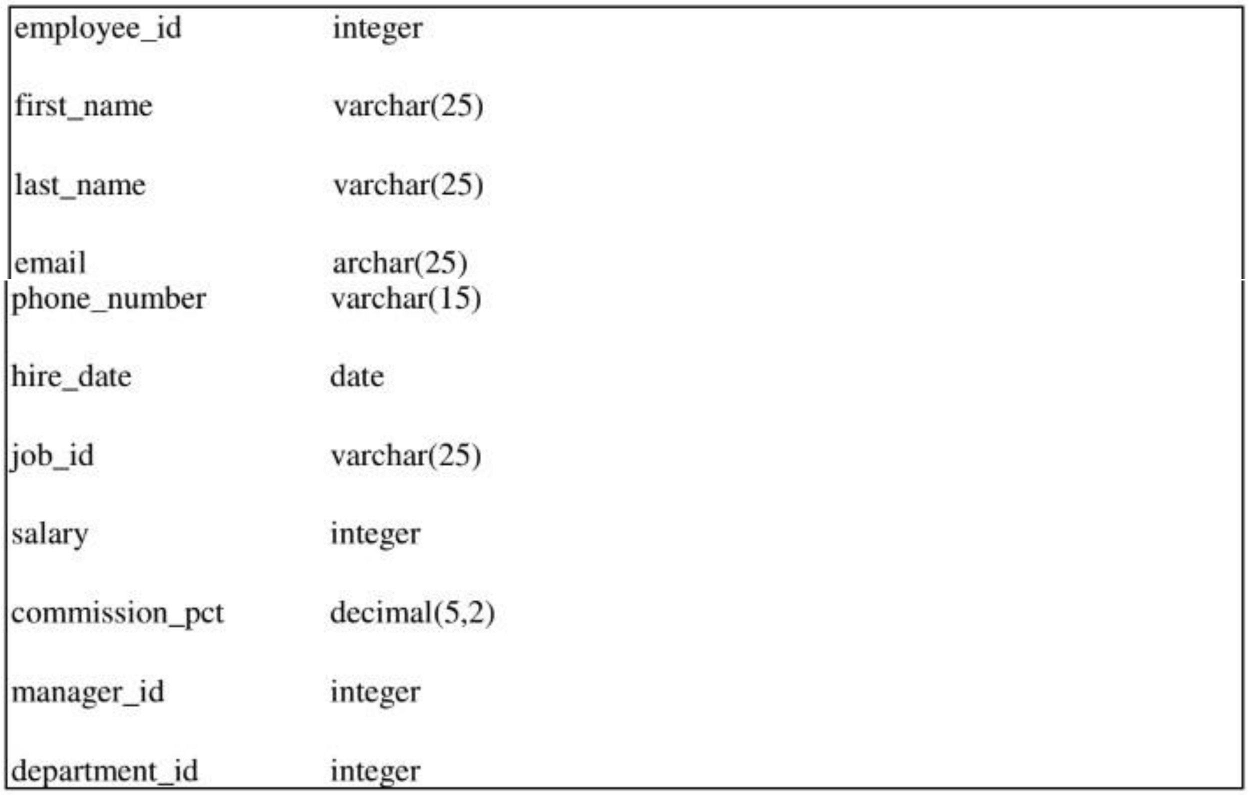


employee_id	integer
first_name	varchar(25)
last_name	varchar(25)
email	archar(25)
phone_number	varchar(15)
hire_date	date
job_id	varchar(25)
salary	integer
commission_pct	decimal(5,2)
manager_id	integer
department_id	integer

```
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



employee_id	integer
first_name	varchar(25)
last_name	varchar(25)
email	archar(25)
phone_number	varchar(15)
hire_date	date
job_id	varchar(25)
salary	integer
commission_pct	decimal(5,2)
manager_id	integer
department_id	integer

```
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(-01, '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.

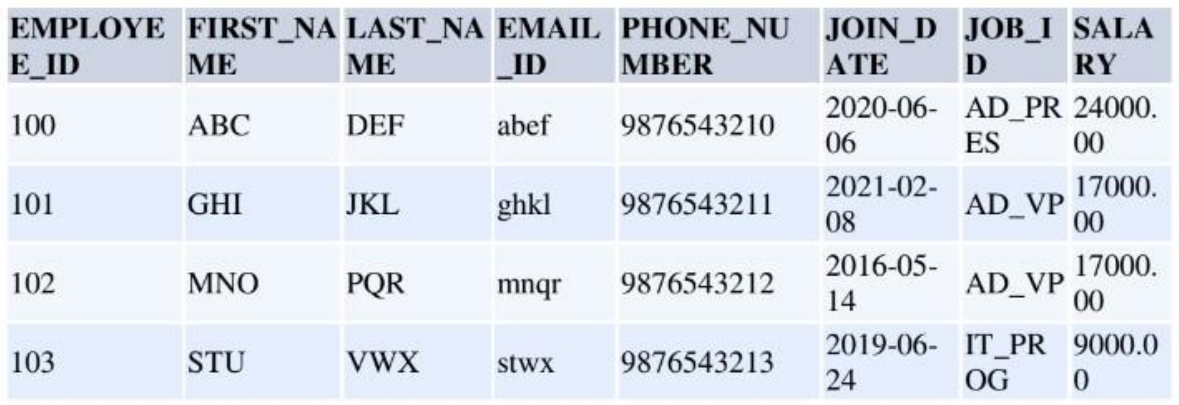
** Consider that we are fetching details of employees from ib_employee table where salary is a parameter for filter.**

```
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 (01, ‘Hemamalani’, 'Elaiyaraja', 'hema243@gmail.com', '6369176255',
TO_DATE('2023-07-12', 'YYYY-MM-DD'), ‘IT’, 23000, 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 (02, 'Hari', 'Dharan', 'haridharan@gmail.com', '9065079091',
TO_DATE('2021-03-02', '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 (03, 'Bharath', 'Sunder', 'bharathsunder107@gmail.com', '7080164590',
TO_DATE('2021-11-24', 'YYYY-MM-DD'), 'FINANCE', 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.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL_ID	PHONE_NUMBER	JOIN_DATE	JOB_ID	SALARY
100	ABC	DEF	abef	9876543210	2020-06-06	AD_PRES	24000.00
101	GHI	JKL	ghkl	9876543211	2021-02-08	AD_VP	17000.00
102	MNO	PQR	mnqr	9876543212	2016-05-14	AD_VP	17000.00
103	STU	VWX	stwx	9876543213	2019-06-24	IT_PROG	9000.00

```
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;
/
```