PL-SQL:

1.Write a PL-SQL Program to create a trigger to backup a row which is delete from the one table and inse rted in the another table.

sol:

```
SQL> -- Create the source_table (if it doesn’t exist)
SQL> CREATE TABLE source_table (
  2      id NUMBER PRIMARY KEY,
  3      name VARCHAR2(100),
  4      age NUMBER
  5  );

Table created.

SQL>
SQL> -- Create the backup_table (if it doesn’t exist)
SQL> CREATE TABLE backup_table (
  2      id NUMBER PRIMARY KEY,
  3      name VARCHAR2(100),
  4      age NUMBER
  5  );

Table created.

SQL>
SQL> -- Create the trigger
SQL> CREATE OR REPLACE TRIGGER backup_trigger
  2  AFTER DELETE ON source_table
  3  FOR EACH ROW
  4  BEGIN
  5      -- Insert the deleted row into the backup_table
  6      INSERT INTO backup_table (id, name, age)
  7      VALUES (:OLD.id, :OLD.name, :OLD.age);
  8
  9      -- Display the backup information
 10       DBMS_OUTPUT.PUT_LINE(’Row with ID ’ || :OLD.id || ’ backed up to backup_table.’);
 11  END;
 12  /

Trigger created.

SQL> -- Enable DBMS_OUTPUT
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> -- Insert some data into the source_table
SQL> INSERT INTO source_table (id, name, age) VALUES (1, ’John Doe’, 25);

1 row created.

SQL> INSERT INTO source_table (id, name, age) VALUES (2, ’Jane Smith’, 30);

1 row created.

SQL>
```

```
SQL> -- Delete a row from the source_table
SQL> DELETE FROM source_table WHERE id = 1;
Row with ID 1 backed up to backup_table.

1 row deleted.

SQL> SELECT * FROM backup_table;

        ID  NAME    AGE
----------------------------------------------------------------------------
         1  John Doe      25
```

2.Write a PL-SQL Program to Print the salary changes when the salary is changed.

sol:

```
SQL> -- Create the employees table (if it doesn't exist)
SQL> CREATE TABLE employees (
  2      employee_id NUMBER PRIMARY KEY,
  3      employee_name VARCHAR2(100),
  4      salary NUMBER,
  5      update_date DATE
  6  );

Table created.

SQL>
SQL> -- Create the trigger
SQL> CREATE OR REPLACE TRIGGER salary_change_trigger
  2  BEFORE UPDATE OF salary ON employees
  3  FOR EACH ROW
  4  BEGIN
  5      -- Check if the salary is being updated
  6      IF :OLD.salary != :NEW.salary THEN
  7          -- Print the salary change information
  8          DBMS_OUTPUT.PUT_LINE('Salary change for Employee ID: ' || :NEW.employee_id);
  9          DBMS_OUTPUT.PUT_LINE('Old Salary: ' || :OLD.salary || ', New Salary: ' || :NEW.salary);
 10          DBMS_OUTPUT.PUT_LINE('Update Date: ' || SYSDATE);
 11      END IF;
 12  END;
 13  /

Trigger created.

SQL> -- Enable DBMS_OUTPUT
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> -- Insert some data into the employees table
SQL> INSERT INTO employees (employee_id, employee_name, salary, update_date) VALUES (1, 'John
Doe', 50000, SYSDATE);

1 row created.

SQL> INSERT INTO employees (employee_id, employee_name, salary, update_date) VALUES (2, 'Jane
Smith', 60000, SYSDATE);
```

```
1 row created.

SQL>
SQL> -- Update the salary for an employee
SQL> UPDATE employees SET salary = 55000 WHERE employee_id = 1;
Salary change for Employee ID: 1
Old Salary: 50000, New Salary: 55000
Update Date: 02-AUG-23

1 row updated.

SQL> select * from employees;

EMPLOYEE_ID   EMPLOYEE_NAME   SALARY UPDATE_DA
-----------------------------------------------------------------------------------------------
         1    John Doe        55000     02-AUG-23
         2    Jane Smith       60000    02-AUG-23
```