Nama : Afina Putri Dayanti
NIM : 825200049
Jurusan : Sistem Informasi
Mata Kuliah : Database Design and Management (Praktikum)

Vocabulary

Identify the vocabulary word for each definition below:

Compound Trigger	is a single trigger that can include actions for each of the four possible timing points
INSTEAD OF Trigger	a trigger which replaces a DML statement on a complex view with DML statements on the tables on which the view is based
Conditional Predicates	predefined Boolean variables INSERTING, DELETING and UPDATING which can be tested in a trigger body to take different code paths depending on which DML statement caused the trigger to fire
:OLD and :NEW Qualifiers	enables a row trigger to access column values in the table row currently being modified by the triggering statement
DML row trigger	a DML trigger which fires once for each row affected by the triggering DML statement

Try It / Solve It

1. Retrieve the code for the AFTER INSERT trigger you created in the previous practice, question 2B. If you have lost the code, here it is again:

```
CREATE OR REPLACE TRIGGER emp_audit_trigg
  AFTER INSERT ON employees
BEGIN
  INSERT INTO audit_table (action) VALUES ('Inserting');
END;
```

Answer :

```
create or replace trigger log_audit_table
after insert on emp1
begin
  insert into audit_table(action, user_name, last_change_date)
  values ('Inserting',user, systimestamp );
end;
```

2. Modify this trigger so that a DELETE on the EMPLOYEES table will fire the same trigger. Use the conditional predicates so an INSERT adds a row to the AUDIT_EMP table with 'Inserted' for the action column and a DELETE adds a row with 'Deleted' in the action column. Save the script and test your trigger by inserting an employee row and then deleting the same row, querying the AUDIT_EMP table each time.

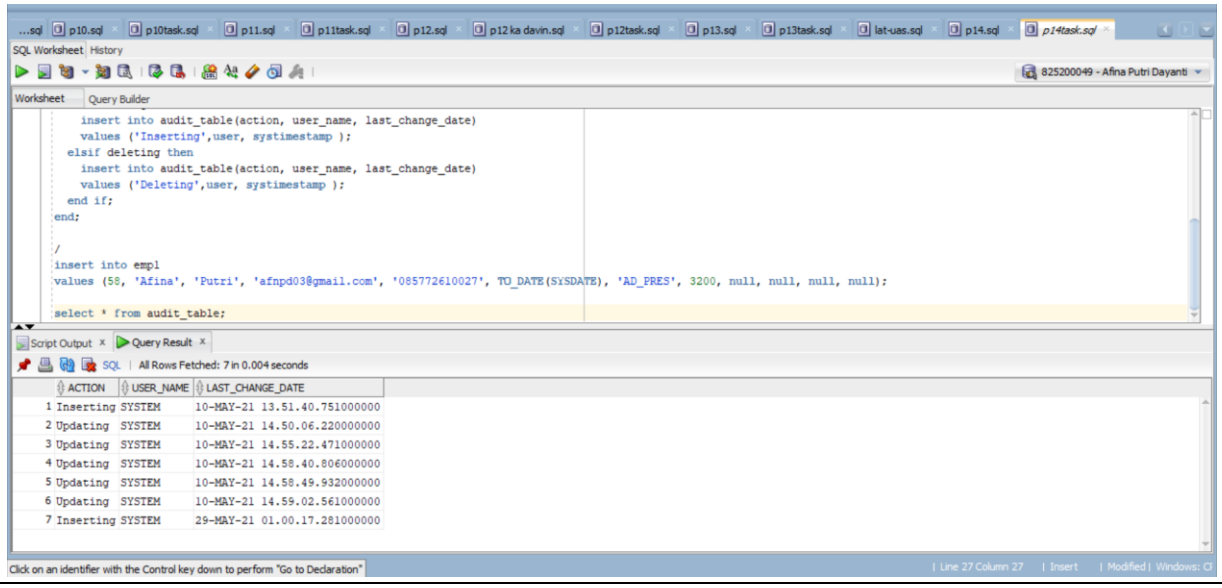
Answer :

```
create or replace trigger log_audit_table
after insert or delete on emp1
```

```
begin
  if inserting then
    insert into audit_table(action, user_name, last_change_date)
    values ('Inserting',user, systimestamp );
  elsif deleting then
    insert into audit_table(action, user_name, last_change_date)
    values ('Deleting',user, systimestamp );
  end if;
end;
```

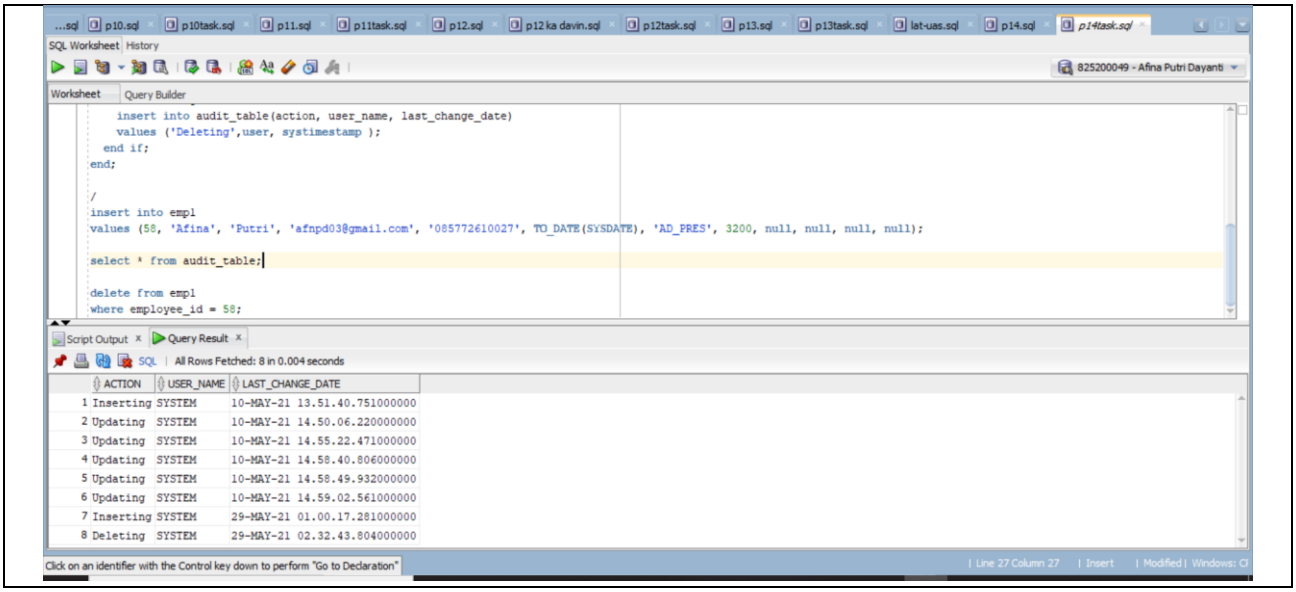
Insert

```
insert into emp1
values (58, 'Afina', 'Putri', 'afnpd03@gmail.com', '085772610027', TO_DATE(SYSDATE), 'AD_PRES',
3200, null, null, null, null);

select * from audit_table;
```



Delete

```
delete from emp1
where employee_id = 58;

select * from audit_table;
```



3. Add a new column called emp_id to the AUDIT_EMP table. This column will contain the employee id of the worker whose record was inserted or deleted. Modify your trigger to be a row trigger so it will fire once for each row affected. The INSERTs into the AUDIT_EMP table should now include the employee id of the affected employee. INSERT and DELETE one or more employees. Query the AUDIT_EMP table to see the audit trail.

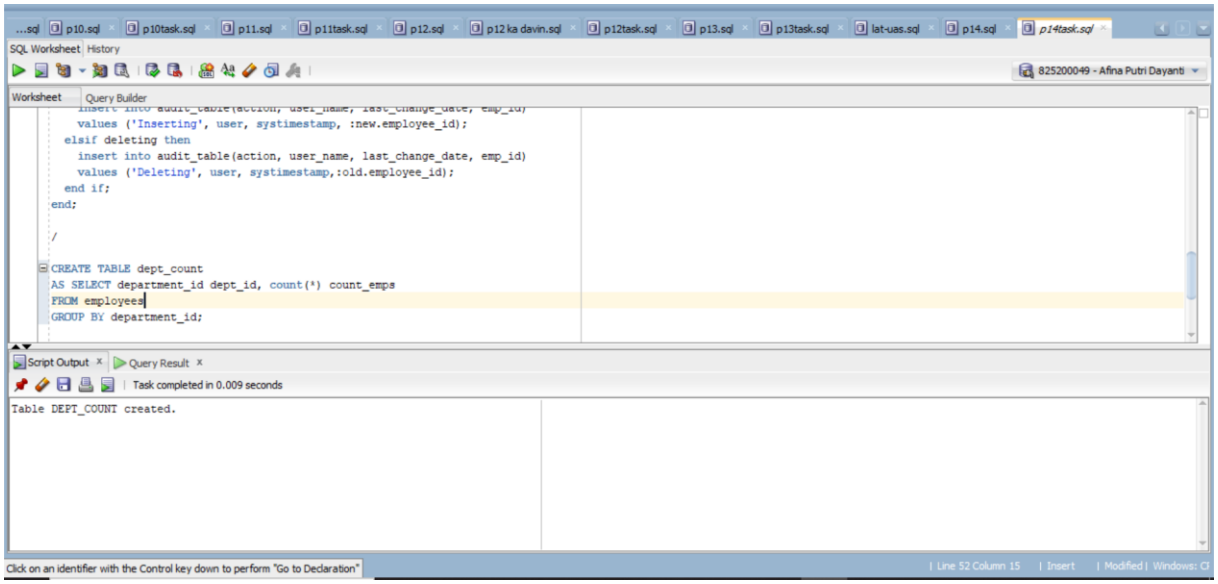
   Answer :

```
alter table audit_table add emp_id number(3);

create or replace trigger log_audit_table
after insert or delete on emp1 for each row
begin
  if inserting then
    insert into audit_table(action, user_name, last_change_date, emp_id)
    values ('Inserting', user, systimestamp, :new.employee_id);
  elsif deleting then
    insert into audit_table(action, user_name, last_change_date, emp_id)
    values ('Deleting', user, systimestamp,:old.employee_id);
  end if;
end;
```

4. To practice using INSTEAD OF triggers, complete the following steps.
   a) Execute the following statement to create a table called DEPT_COUNT that keeps track of how many employees are in each department.

      CREATE TABLE dept_count
      AS SELECT department_id dept_id, count(*) count_emps
      FROM employees
      GROUP BY department_id;

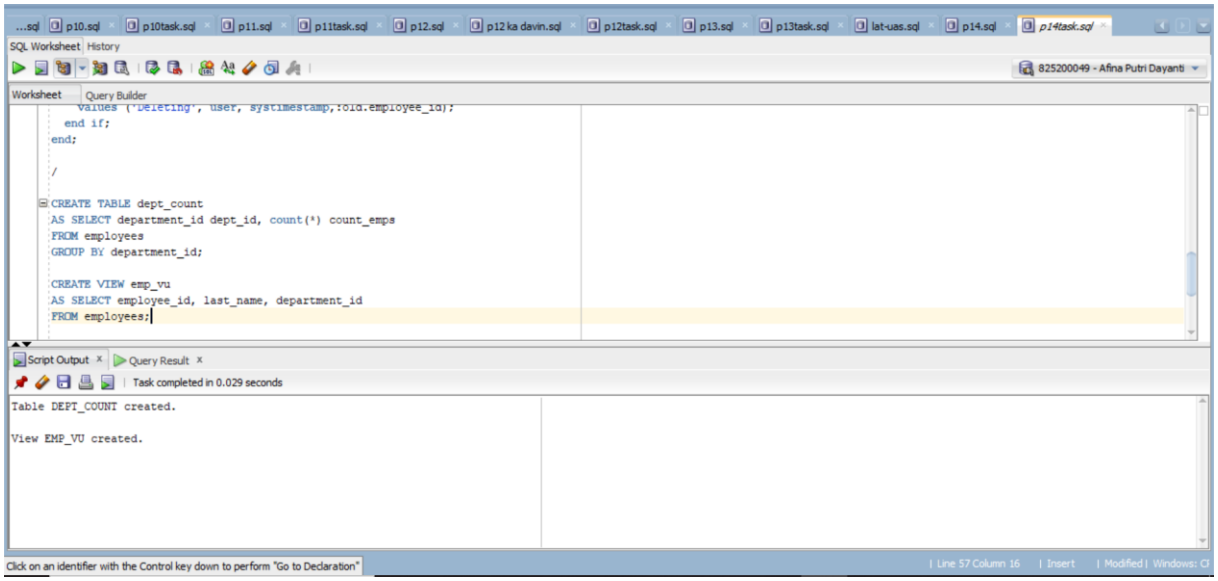
      Answer :



b) Execute the following statement to create a view of the EMPLOYEES table called EMP_VU.

```
CREATE VIEW emp_vu
AS SELECT employee_id, last_name, department_id
FROM employees;
```

Answer :



c) Create an INSTEAD OF row trigger on EMP_VU that increases the current count for a department by 1 if a new employee is added and subtracts 1 from the count for a department if an employee is deleted.

Answer :

```
create or replace trigger count_trigger
instead of insert or delete on emp_vu
begin
  if inserting then
    update dept_count set count_emps = count_emps + 1
```

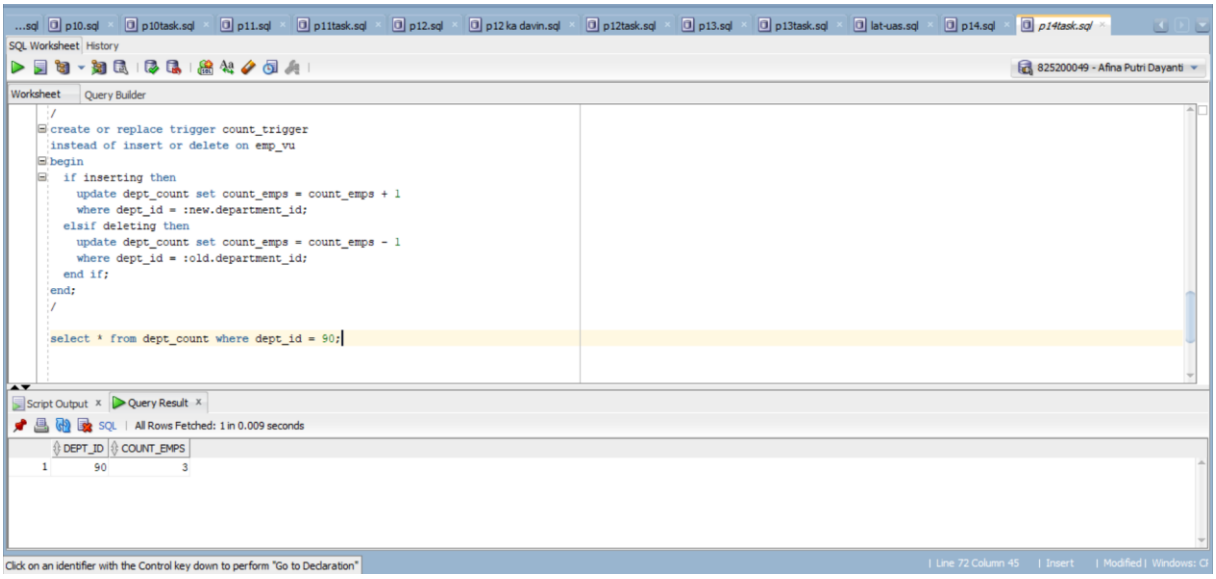
```
    where dept_id = :new.department_id;
  elsif deleting then
    update dept_count set count_emps = count_emps - 1
    where dept_id = :old.department_id;
  end if;
end;
```

d) Look at the counts for all departments in DEPT_COUNT. Test to see if your trigger fires correctly by inserting a row into EMP_VU. Look at the count for the department of the new employee. Delete a row from EMP_VU. Look at the count for the department where the employee was just deleted.

Answer :

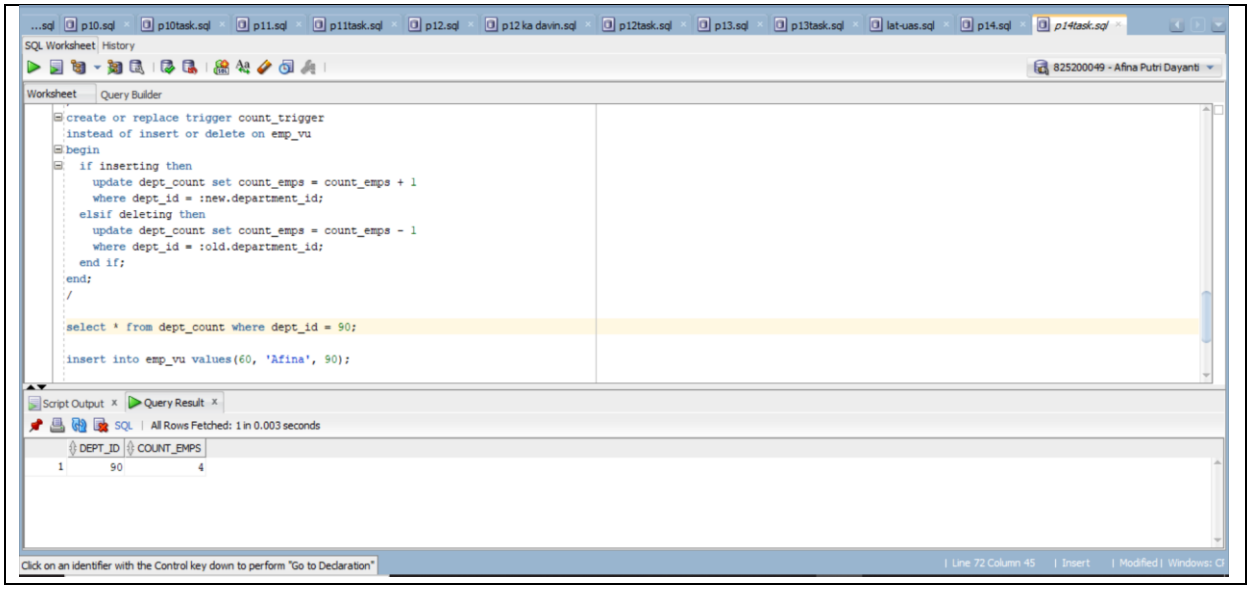
Before Insert

```
select * from dept_count where dept_id = 90;
```



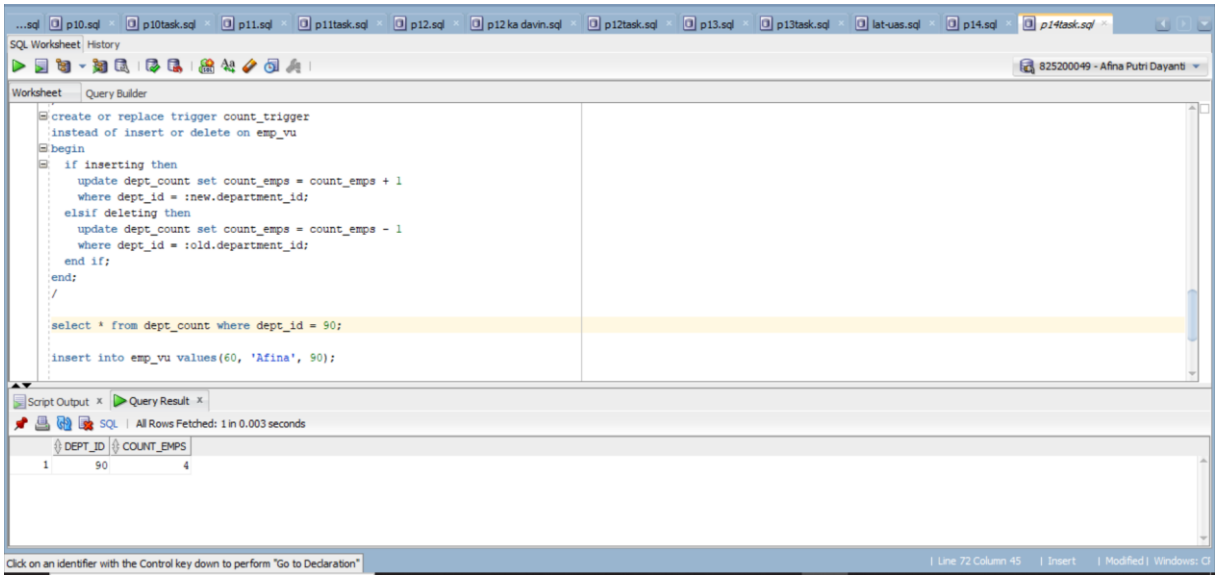
After Insert

```
insert into emp_vu values(60, 'Afina', 90);
select * from dept_count where dept_id = 90;
```



Before Delete

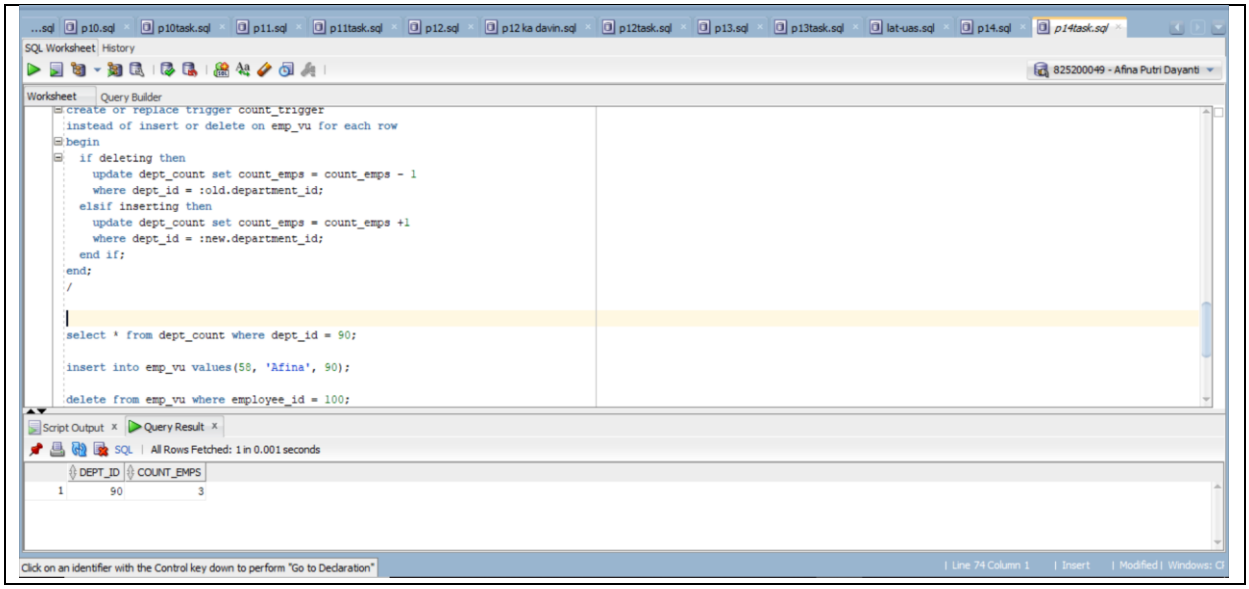
select * from dept_count where dept_id = 90;



After Delete

delete from emp_vu where employee_id = 100;

select * from dept_count where dept_id = 90;



5. In this question, you will create a compound trigger. Once again, you will use the AUDIT_TABLE you created in a previous exercise. If you have lost that table, below is the code to recreate it.

```
CREATE TABLE audit_table
(action              VARCHAR2(50),
user_name            VARCHAR2(30) DEFAULT USER,
last_change_date     TIMESTAMP DEFAULT SYSTIMESTAMP,
emp_id               NUMBER(6));
```

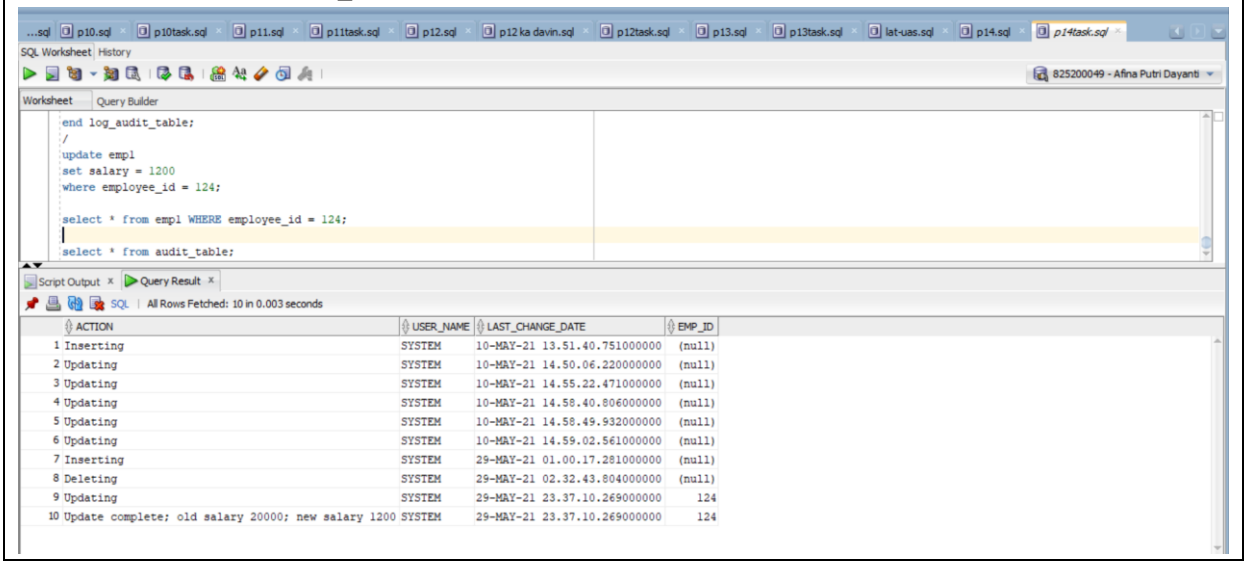
a) Create a compound trigger emp_audit_trigg on the EMPLOYEES table for the following events: when updating the salary column of the EMPLOYEES table, enter the value 'Updating' into the action column of the AUDIT_TABLE before the change occurs. Next, once the action is complete, change the action to 'Update complete; old salary was (old_sal); new salary is (new_sal)' where old_sal is the original salary before the UPDATE, and new_sal is the new salary.

Answer :

```
create or replace trigger log_audit_table
for update of salary on emp1 compound trigger
  log varchar2(200);
  before each row is begin
    insert into audit_table(action, user_name, last_change_date, emp_id)
    values ('Updating', user, systimestamp, :new.employee_id);
  end before each row;
  after each row is begin
    log := 'Update complete; old salary ' || to_char(:old.salary) || '; new salary ' ||
to_char(:new.salary);
    insert into audit_table(action, user_name, last_change_date, emp_id)
    values (log, user, systimestamp, :new.employee_id);
  end after each row;
end log_audit_table;
```

b) Test your trigger by updating the salary of employee_id = 124 to be 1200, then querying the AUDIT_TABLE to see that it contains a new row.

```
update emp1
set salary = 1200
where employee_id = 124;

select * from audit_table;
```



	ACTION	USER_NAME	LAST_CHANGE_DATE	EMP_ID
1	Inserting	SYSTEM	10-MAY-21 13.51.40.751000000	(null)
2	Updating	SYSTEM	10-MAY-21 14.50.06.220000000	(null)
3	Updating	SYSTEM	10-MAY-21 14.55.22.471000000	(null)
4	Updating	SYSTEM	10-MAY-21 14.58.40.806000000	(null)
5	Updating	SYSTEM	10-MAY-21 14.58.49.932000000	(null)
6	Updating	SYSTEM	10-MAY-21 14.59.02.561000000	(null)
7	Inserting	SYSTEM	29-MAY-21 01.00.17.281000000	(null)
8	Deleting	SYSTEM	29-MAY-21 02.32.43.804000000	(null)
9	Updating	SYSTEM	29-MAY-21 23.37.10.269000000	124
10	Update complete; old salary 20000; new salary 1200	SYSTEM	29-MAY-21 23.37.10.269000000	124