

The Islamic University of Gaza  
Engineering Faculty  
Computer Engineering Department



# Advance Database Lab

## Oracle Project Part 3

**Done By :**

Ayah Abu Hamra

220064486

Islam Abu Salem

220060769

**Submitted To :**

Eng. Doa'a Abu Jabal

For your college database created in part 1&2:

- Create a Package college that will contain all your work .

Package specification:-

```
create or replace package college is

PROCEDURE Teachers_insert(id number,
First_Name varchar2,
Last_name varchar2,
Head_ID Number);
-----

PROCEDURE Courses_insert(course_ID Number,
Course_Name varchar2,
course_year Number,
Number_Of_Hours Number,
Description varchar2,
Teacher_id number);
-----

Procedure departments_heads_insert(Department varchar2,
Head_ID NUMBER,
Head_Assisst_ID Number);
-----

PROCEDURE students_insert(ID number,
First_Name varchar2,
Last_name varchar2,
course_ID number,
course_Year number,
number_of_hours number,
Grade number);
-----

PROCEDURE Teacher_Courses_insert2(Teacher_id number,
                                Teacher_Name varchar2,
                                head number,
                                Course_code varchar2,
                                Course_Name varchar2,
                                Course_description varchar2);
-----

PROCEDURE Student_Courses_insert2(stu_id number,
                                Teacher_id number,
                                head number,
                                Student_Name varchar2,
                                Teacher_Name varchar2,
                                Course_code varchar2,
                                Course_Name varchar2,
                                Course_description varchar2,
                                grade number);
-----

PROCEDURE Tea_Course_Students_insert2(Teacher_id number,
                                Teacher_Name varchar2,
                                Course_code varchar2,
                                course_name varchar2,
                                stu_id number,
                                Student_Name varchar2,
                                grade number);
-----

PROCEDURE Degree_update(New_Degree Number, Student_ID Number);
-----

PROCEDURE Teacher_update(teach_id number, id number);
-----

PROCEDURE Head_id_update(new_ID NUMBER, dep VARCHAR2);
-----

PROCEDURE Delete_teacher(teach_first_name varchar2,
teach_last_name varchar2);
-----

PROCEDURE Print_Teacher_Name;
-----

PROCEDURE Print_Students(ID NUMBER, cYear NUMBER, Hours NUMBER);
-----
```

```
PROCEDURE Print_Students(Name varchar2);  
-----  
FUNCTION Teacher_info(teach_id number) RETURN VARCHAR2;  
-----  
FUNCTION Course_description(code VARCHAR2) RETURN VARCHAR2;  
-----  
  
end college;
```

Package Body:-

```

SET SERVEROUTPUT ON

create or replace package body college is

PROCEDURE Teachers_insert(id number,
                          First_Name varchar2,
                          Last_name varchar2,
                          Head_ID Number) IS

begin
insert into Teachers Values (id, First_Name, Last_name, Head_ID);
end;

-----
PROCEDURE Courses_insert(course_ID Number,
                          Course_Name varchar2,
                          course_year Number,
                          Number_Of_Hours Number,
                          Description varchar2,
                          Teacher_id number) IS

begin
insert into Courses
values
(course_ID,
 Course_Name,
 course_year,
 Number_Of_Hours,
 Description,
 Teacher_id);

end;

-----
Procedure departments_heads_insert(Department varchar2,
                                    Head_ID NUMBER,
                                    Head_Assist_ID Number) IS

begin
begin
insert into departments_heads
values
(Department, Head_ID, Head_Assist_ID);
end;

-----
PROCEDURE students_insert(ID number,
                          First_Name          varchar2,
                          Last_name          varchar2,
                          course_ID          number,
                          course_Year       number,
                          number_of_hours   number,
                          Grade             number) IS

begin
insert into students
values
(ID,
 First_Name,
 Last_name,
 course_ID,
 course_Year,
 number_of_hours,
 Grade);

end;

-----
PROCEDURE Teacher_Courses_insert2(Teacher_id number,
                                   Teacher_Name  varchar2,
                                   head number,
                                   course_code    varchar2,
                                   Course_Name    varchar2,
                                   Course_description varchar2) IS

indef2 number;
Tfname Teachers.First_Name$type;

```

```

Tlname Teachers.Last_Name&type
cid courses.course_id&type
cyear courses.Course_Year&type
chours courses.Number_Of_Hours&type

```

```

begin

```

```

index2:=instr(Teacher_Name, ' ');
Tfname:=substr(Teacher_Name,1,index2);
Tlname:=substr(Teacher_Name,index2);

```

```

cid:=substr(Course_code,1,1);
cyear:=substr(Course_code,2,1);
chours:=substr(Course_code,3,1);

```

```

insert into teachers

```

```

values (Teacher_id,
        Tfname,
        Tlname,
        head);

```

```

insert into courses

```

```

Values
    (cid,
     Course_Name,
     cyear,
     chours,
     Course_description,
     Teacher_id);

```

```

end;
-----|

```

```

PROCEDURE Student_Courses_insert2(stu_id number,
                                   Teacher_id number,
                                   head number,
                                   Student_Name          varchar2,
                                   Teacher_Name           varchar2,
                                   Course_code            varchar2,
                                   Course_Name            varchar2,
                                   Course_description     varchar2,
                                   grade                  number) IS

```

```

index1 number;
index2 number;
fname Students.First_Name&type
lname Students.Last_Name&type
Tfname Teachers.First_Name&type
Tlname Teachers.Last_Name&type
cid courses.course_id&type
cyear courses.Course_Year&type
chours courses.Number_Of_Hours&type

```

```

begin

```

```

index1:=instr(Student_Name, ' ');
fname:=substr(Student_Name,1,index1);
lname:=substr(Student_Name,index1);
index2:=instr(Teacher_Name, ' ');
Tfname:=substr(Teacher_Name,1,index2);
Tlname:=substr(Teacher_Name,index2);

```

```

cid:=substr(Course_code,1,1);
cyear:=substr(Course_code,2,1);
chours:=substr(Course_code,3,1);

```

```

insert into teachers

```

```

values (Teacher_id,

```

```

values(Teacher_id,
       Tfname,
       Tlname,
       head);

insert into courses
Values
(cid,
 Course_Name,
 cyear,
 chours,
 Course_description,
 Teacher_id);

insert into Students
values
(stu_id,
 fname,
 lname,
 cid,
 cyear,
 chours,
 grade);

end;
-----
PROCEDURE Tea_Course_Students_insert2(Teacher_id number,
                                     Teacher_Name varchar2,
                                     Course_code varchar2,
                                     course_name varchar2,
                                     stu_id number,
                                     Student_Name varchar2,
                                     grade number) IS

```

```

index1 number;
index2 number;
fname Students.First_Name%type;
lname Students.Last_Name%type;
Tfname Teachers.First_Name%type;
Tlname Teachers.Last_Name%type;
cid courses.course_id%type;
cyear courses.Course_Year%type;
chours courses.Number_of_Hours%type;

BEGIN
index1:=instr(Student_Name,' ');
fname:=substr(Student_Name,1,index1);
lname:=substr(Student_Name,index1+1);
index2:=instr(Teacher_Name,' ');
Tfname:=substr(Teacher_Name,1,index2);
Tlname:=substr(Teacher_Name,index2+1);

cid:=substr(Course_code,1,1);
cyear:=substr(Course_code,2,1);
chours:=substr(Course_code,3,1);

insert into teachers(id,first_name,last_name)
values(Teacher_id,
       Tfname,
       Tlname
       );

insert into courses(course_id,course_name,course_year,number_of_hours,teacher_id)
Values
(cid,
 course name,
 cyear,
 chours,

```

```

        Teacher_id);

insert into Students
values
    (stu_id,
     fname,
     lname,
     cid,
     cyear,
     chours,
     grade);

END;
-----
PROCEDURE Degree_update(New_Degree Number, Student_ID Number) IS
BEGIN
    Update Students set grade = New_Degree where ID = Student_ID;
end;
-----
PROCEDURE Teacher_update(teach_id number, id number) is
BEGIN
    update Courses set Teacher_id = teach_id where course_ID = id;
END;
-----
PROCEDURE Head_id_update(new_ID NUMBER, dep VARCHAR2) is
    head departments_heads.head_ID%TYPE;
BEGIN
    SELECT head_ID into head FROM departments_heads WHERE Department = dep;
    DBMS_OUTPUT.PUT_LINE(head);

    update departments_heads set head_ID = NEW_ID where Department = dep;

END;
-----
PROCEDURE Delete_teacher(teach_first_name varchar2,
                        teach_last_name varchar2) IS
    teach_F TEACHERS.FIRST_NAME%TYPE;
    teach_L TEACHERS.LAST_NAME%TYPE;

BEGIN
    SELECT FIRST_NAME, LAST_NAME INTO teach_F, teach_L
    FROM TEACHERS
    WHERE FIRST_NAME = teach_first_name
    and LAST_NAME = teach_last_name;
    DELETE FROM TEACHERS
    WHERE FIRST_NAME = teach_first_name
    and LAST_NAME = teach_last_name;
    DBMS_OUTPUT.PUT_LINE(teach_F);
    DBMS_OUTPUT.PUT_LINE(teach_L);
END;
-----
PROCEDURE Print_Teacher_Name IS
    teach_F TEACHERS.FIRST_NAME%TYPE;
    teach_L TEACHERS.LAST_NAME%TYPE;
    counter number;
BEGIN
    counter:=0;
    SELECT First_Name, Last_Name
    INTO teach_F, teach_L
    FROM Teachers
    WHERE Head_ID IN (SELECT Head_ID FROM departments_heads);
    while counter< SQL%rowcount loop
        counter:=counter+1;
        DBMS_OUTPUT.PUT_LINE(teach_F);
        DBMS_OUTPUT.PUT_LINE(teach_L);
    end loop;
end;

```

```

PROCEDURE Print_Students(ID NUMBER, cYear NUMBER, Hours NUMBER) IS
    std_F Students.FIRST_NAME%TYPE;
    std_L Students.LAST_NAME%TYPE;
    grd Students.Grade%TYPE;
    counter number;
    Cursor students_cursor is
    SELECT First_Name, Last_Name, GRADE
        FROM Students
        WHERE Course_ID = ID
            and Course_Year = cYear
            and Number_Of_Hours = Hours;

```

```

BEGIN
open students_cursor;
counter:=0;
while counter<students_cursor%rowcount loop
fetch students_cursor into std_F,std_L,grd;
DBMS_OUTPUT.PUT_LINE(std_F);
DBMS_OUTPUT.PUT_LINE(std_L);
DBMS_OUTPUT.PUT_LINE(grd);
counter:=counter+1;
end loop;

```

```

END;

```

```

PROCEDURE Print_Students(Name varchar2) IS
    std_name Teacher_Course_Students."Student Name"%type;
    std_cr Teacher_Course_Students."Course Code"%type;
    std_grd Teacher_Course_Students.Grade%type;
    counter number;
    Cursor students_cursor is
    SELECT "Student Name", "Course Code", Grade
        FROM Teacher_Course_Students
        WHERE "Teacher Name" = Name;

```

```

BEGIN
open students_cursor;
counter:=0;
while counter<students_cursor%rowcount loop
fetch students_cursor into std_name,std_cr,std_grd;
DBMS_OUTPUT.PUT_LINE(std_name);
DBMS_OUTPUT.PUT_LINE(std_cr);
DBMS_OUTPUT.PUT_LINE(std_grd);
counter:=counter+1;
end loop;
END;

```

```

FUNCTION Teacher_info(teach_id number) RETURN VARCHAR2 IS
first TEACHERS.FIRST_NAME%TYPE;
last TEACHERS.LAST_NAME%TYPE;

```

```

BEGIN
SELECT FIRST_NAME, LAST_NAME
INTO first, last
FROM TEACHERS
WHERE ID = teach_id;
DBMS_OUTPUT.PUT_LINE(first || ' ' || last);
RETURN first || ' ' || last;
END;

```

```

FUNCTION Course_description(code VARCHAR2) RETURN VARCHAR2 IS
    des Teacher_Courses.Description%TYPE;
    id number(1);
    cyear number(1);
    hours number(1);

```

```

BEGIN
id:=substr(code,1,1);
cyear:=substr(code,2,1);

```



```
hours:=substr(code,3,1);

SELECT Description
INTO des
FROM Courses
WHERE Course_id=id and
Course_year=cyear and
Number_of_Hours=hours;
DEMS_OUTPUT.PUT_LINE( The description of the course ccda '||code| ' is '||des);
RETURN des;
END;
-----

Begin
college.Teachers_insert(12,'Islam','Salem',6);
end collage;
```

- Procedures Test :-
  - Teachers\_insert

SQL Window - select \* from Teache...

SQL Output Statistics

```
select * from Teachers;
```

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	6
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6

1:24 6 rows selected in 0.094 seconds

Before calling "Teacher\_insert" procedure.

SQL Window - Begin college.Teachers\_i...

SQL Output Statistics

```
Begin
college.Teachers_insert(15, 'Ayah', 'Ahmed', 6);
end;
```

3:5 Done in 0 seconds

calling "Teacher\_insert" procedure.

SQL Window - select \* From teachers;

SQL Output Statistics

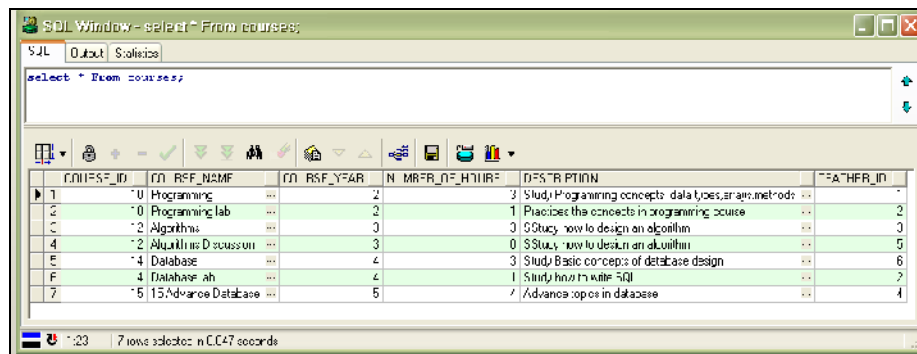
```
select * From teachers;
```

	ID	FRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	6
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	15	Ayah	Ahmed	6

1R id, number(8), mandatory

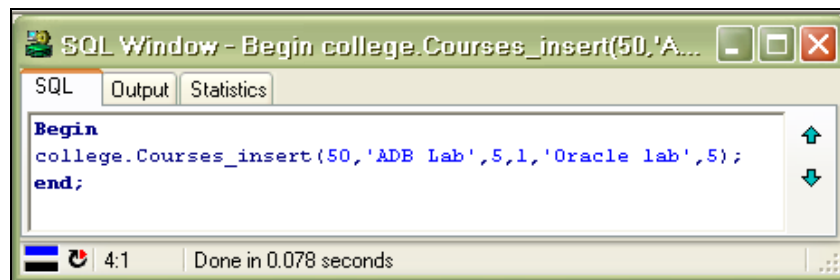
After calling "Teacher\_insert" procedure.

## 2. Courses\_insert



COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	10 Programming	2	3	Study Programming concepts: data types, arrays, methods	1
2	10 Programming lab	2	1	Practice the concepts in programming course	2
3	12 Algorithms	3	3	Study how to design an algorithm	3
4	12 Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14 Database	4	3	Study Basic concepts of database design	6
6	14 Database lab	4	1	Study how to write SQL	2
7	15 Advance Database	5	4	Advance topics in database	4

before calling "course\_insert" procedure.



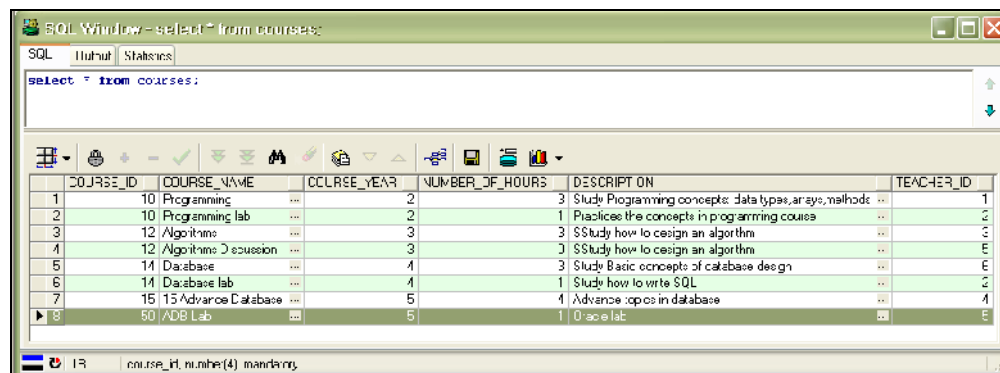
```

Begin
college.Courses_insert(50,'ADB Lab',5,1,'Oracle lab',5);
end;

```

Done in 0.078 seconds

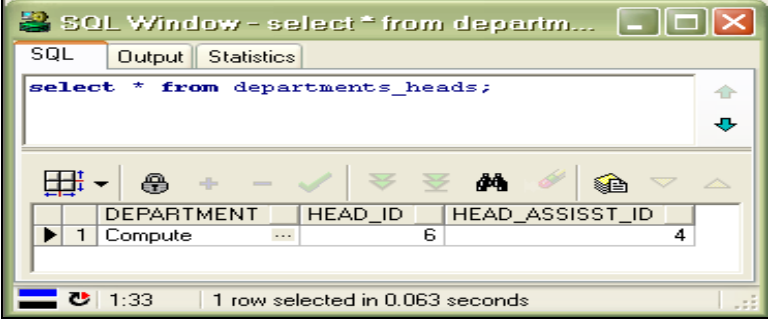
calling " course \_insert" procedure.



COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	10 Programming	2	3	Study Programming concepts: data types, arrays, methods	1
2	10 Programming lab	2	1	Practice the concepts in programming course	2
3	12 Algorithms	3	3	Study how to design an algorithm	3
4	12 Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14 Database	4	3	Study Basic concepts of database design	6
6	14 Database lab	4	1	Study how to write SQL	2
7	15 Advance Database	5	4	Advance topics in database	4
8	50 ADB Lab	5	1	Oracle lab	5

After calling " course \_insert" procedure.

3. departments\_heads\_inser  
t



SQL Window - select \* from departm...

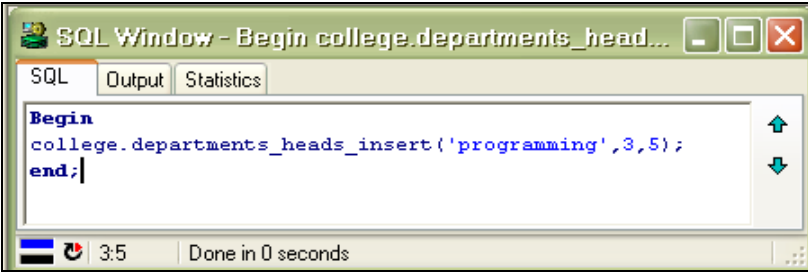
SQL Output Statistics

```
select * from departments_heads;
```

	DEPARTMENT	HEAD_ID	HEAD_ASSISST_ID
1	Compute	6	4

1:33 1 row selected in 0.063 seconds

Before calling "department\_heads\_insert" procedure.



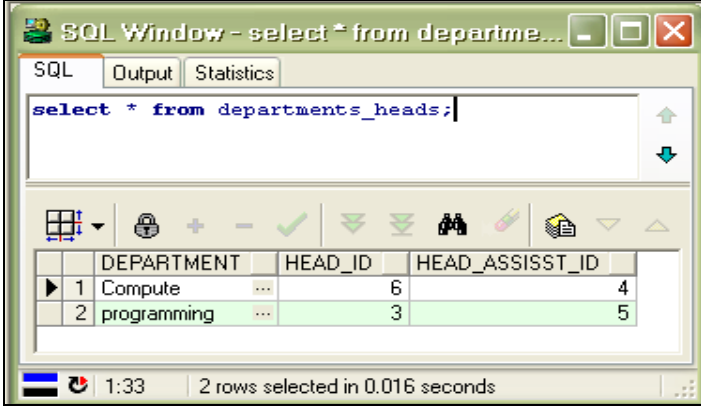
SQL Window - Begin college.departments\_head...

SQL Output Statistics

```
Begin  
college.departments_heads_insert('programming',3,5);  
end;
```

3:5 Done in 0 seconds

calling " department\_heads \_insert" procedure.



SQL Window - select \* from departme...

SQL Output Statistics

```
select * from departments_heads;
```

	DEPARTMENT	HEAD_ID	HEAD_ASSISST_ID
1	Compute	6	4
2	programming	3	5

1:33 2 rows selected in 0.016 seconds

After calling " department\_heads \_insert" procedure.

#### 4. students\_insert

SQL Window - select \* from students;

SQL Output Statistics

```
select * from students;
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	15	5	4	92
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88

1:24 6 rows selected in 0.062 seconds

SQL Window - begin college.students\_insert(8,'Islam',...

SQL Output Statistics

```
begin  
college.students_insert(8,'Islam','Abu-Salem',12,3,3,95);  
end;
```

2:58 Done in 0.031 seconds

SQL Window - select \* from students;

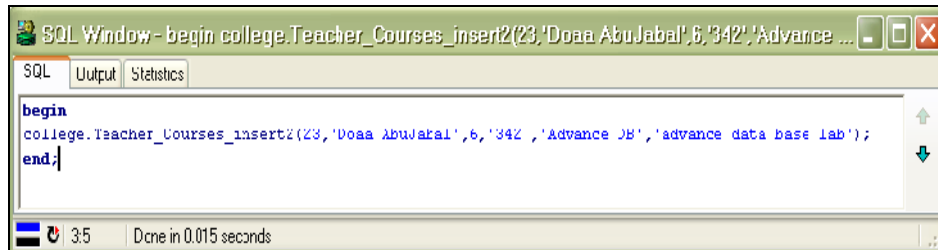
SQL Output Statistics

```
select * from students;
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	15	5	4	92
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88
7	8	Islam	Abu-Salem	12	3	3	95

1R id, number(8), mandatory

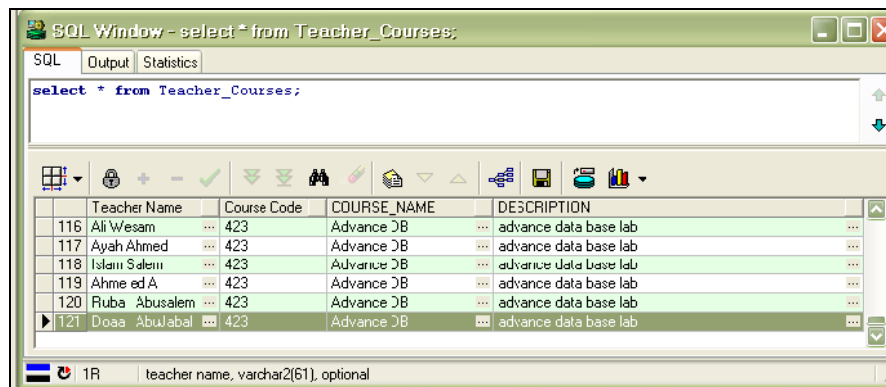
## 5. Teacher\_Courses\_insert2



The screenshot shows an SQL Window titled "SQL Window - begin college.Teacher\_Courses\_insert2(23,'Doaa AbuJabal',6,'342','Advance ...". The window has tabs for "SQL", "Output", and "Statistics". The SQL tab is active, displaying the following code:

```
begin
college.Teacher_Courses_insert2(23,'Doaa AbuJabal',6,'342','Advance DB','advance data base lab');
end;
```

At the bottom of the window, a status bar indicates "3:5 Done in 0.015 seconds".



The screenshot shows an SQL Window titled "SQL Window - select \* from Teacher\_Courses;". The window has tabs for "SQL", "Output", and "Statistics". The SQL tab is active, displaying the following code:

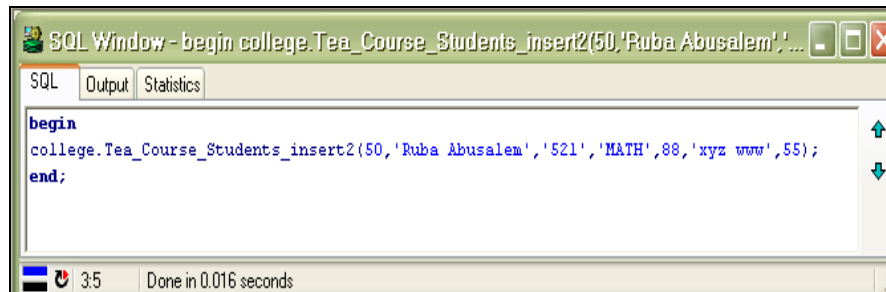
```
select * from Teacher_Courses;
```

Below the code, there is a toolbar with various icons. Below the toolbar, a table displays the results of the query. The table has five columns: "Teacher Name", "Course Code", "COURSE\_NAME", and "DESCRIPTION". The table contains six rows of data, with the last row highlighted in green.

	Teacher Name	Course Code	COURSE_NAME	DESCRIPTION
116	Ali Wesam	423	Advance DB	advance data base lab
117	Ayah Ahmed	423	Advance DB	advance data base lab
118	Islam Salem	423	Advance DB	advance data base lab
119	Ahmed A	423	Advance DB	advance data base lab
120	Ruba Abusaleh	423	Advance DB	advance data base lab
121	Doaa AbuJabal	423	Advance DB	advance data base lab

At the bottom of the window, a status bar indicates "1R teacher name, varchar2(61), optional".

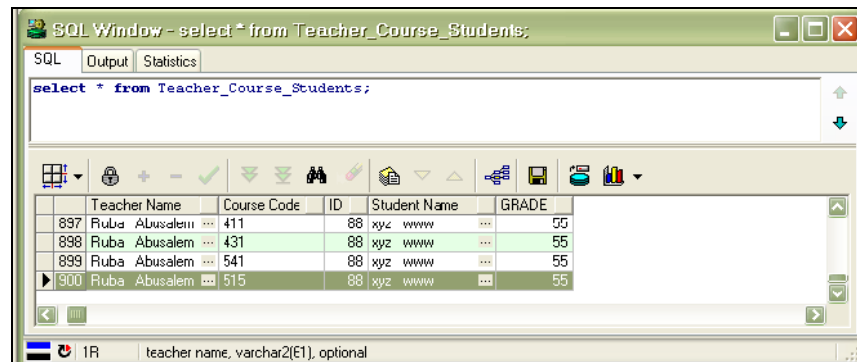
## 6. Tea\_Course\_Students\_insert2



The screenshot shows an SQL Window titled "SQL Window - begin college.Tea\_Course\_Students\_insert2(50,'Ruba Abusalem','...". The window has tabs for "SQL", "Output", and "Statistics". The SQL tab is active, displaying the following code:

```
begin
college.Tea_Course_Students_insert2(50,'Ruba Abusalem','521','MATH',88,'xyz www',55);
end;
```

At the bottom of the window, the status bar indicates "3:5" and "Done in 0.016 seconds".



The screenshot shows an SQL Window titled "SQL Window - select \* from Teacher\_Course\_Students;". The window has tabs for "SQL", "Output", and "Statistics". The SQL tab is active, displaying the following code:

```
select * from Teacher_Course_Students;
```

Below the code, there is a toolbar with various icons. Below the toolbar, a table displays the results of the query. The table has six columns: Teacher Name, Course Code, ID, Student Name, and GRADE. The data is as follows:

Teacher Name	Course Code	ID	Student Name	GRADE
897 Ruba Abusalem	411	88	xyz www	55
898 Ruba Abusalem	431	88	xyz www	55
899 Ruba Abusalem	541	88	xyz www	55
900 Ruba Abusalem	515	88	xyz www	55

At the bottom of the window, the status bar indicates "1R" and "teacher name, varchar2(51), optional".

## 7. Degree\_update

SQL Window - select \* from students;

SQL Output Statistics

```
select * from students;
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	15	5	4	92
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88
7	8	Islam	Abu Solom	12	3	3	95
0	00	Isra	aa Galem	3	1	3	90

1:24 0 rows selected in 0.016 seconds

SQL Window - Begin college....

SQL Output Statistics

```
Begin
college.Degree_update(60,2);
end;
```

3:5 Done in 0 seconds

SQL Window - select \* from students;

SQL Output Statistics

```
select * from students;
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	15	5	4	60
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88
7	8	Islam	Abu-Salem	12	3	3	95
8	80	Isra	aa Salem	3	1	3	99

1R id, number(8), mandatory



## 8. Teacher\_update

SQL Window - select \* from courses;

SQL Output Statistics

```
select * from courses;
```

	COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	13	Programming	2	3	Study Programming concepts: data types, arrays, methods	1
2	13	Programming lab	2	1	Practices the concepts in programming course	2
3	12	Algorithms	3	3	Study how to design an algorithm	3
4	12	Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14	Database	4	3	Study Basic concepts of database design	6
6	14	Database lab	4	1	Study how to write SQL	2
7	15	Advance Database	5	4	Advance topics in database	4
8	50	ADB Lab	5	1	Oracle lab	5
9	3	OS	1	3	os os	90

1/23 Rows selected in 0.315 seconds

SQL Window - begin college.Teacher\_...

SQL Output Statistics

```
begin
college.Teacher_update(1,3);
end;
```

3:5 Done in 0.109 seconds

SQL Window - select \* from courses;

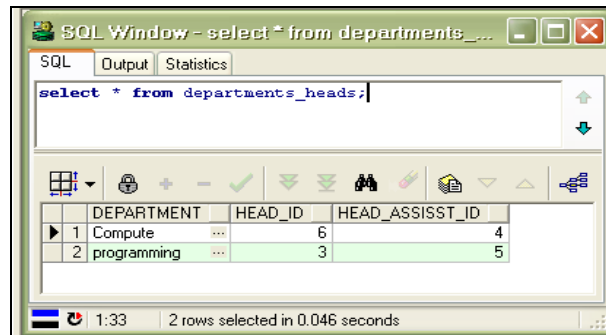
SQL Output Statistics

```
select * from courses;
```

	COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	13	Programming	2	3	Study Programming concepts: data types, arrays, methods	1
2	13	Programming lab	2	1	Practices the concepts in programming course	2
3	12	Algorithms	3	3	Study how to design an algorithm	3
4	12	Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14	Database	4	3	Study Basic concepts of database design	6
6	14	Database lab	4	1	Study how to write SQL	2
7	15	Advance Database	5	4	Advance topics in database	4
8	50	ADB Lab	5	1	Oracle lab	5
9	3	OS	1	3	os os	90

1/23 Rows selected in 0.315 seconds

## 9. Head\_id\_update

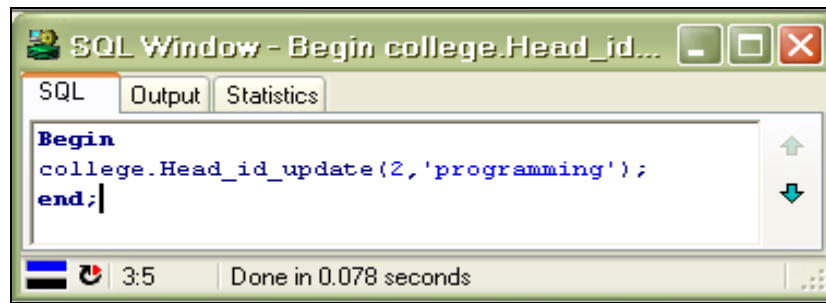


SQL Window - select \* from departments\_...

```
select * from departments_heads;
```

	DEPARTMENT	HEAD_ID	HEAD_ASSIST_ID
1	Compute	6	4
2	programming	3	5

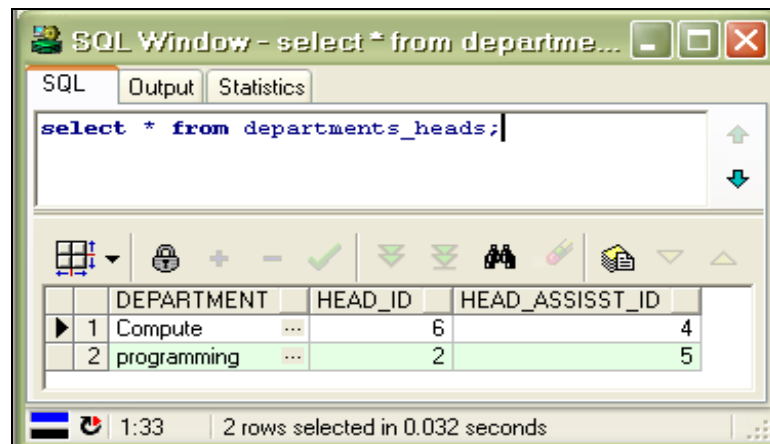
1:33 2 rows selected in 0.046 seconds



SQL Window - Begin college.Head\_id...

```
Begin  
college.Head_id_update(2,'programming');  
end;
```

3:5 Done in 0.078 seconds



SQL Window - select \* from departme...

```
select * from departments_heads;
```

	DEPARTMENT	HEAD_ID	HEAD_ASSIST_ID
1	Compute	6	4
2	programming	2	5

1:33 2 rows selected in 0.032 seconds

## 10. Delete\_teacher

SQL Window - select \* from teachers;

```
select * from teachers;
```

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	6
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	15	Ayah	Ahmed	6
8	12	Islam	Salem	6
9	90	Ahmed	ed A	6

1:24 9 rows selected in 0.053 seconds

SQL Window - begin college.Delete...

```
begin  
college.Delete_teacher('Islam','Salem');  
end;
```

2:41 Done in 0.015 seconds

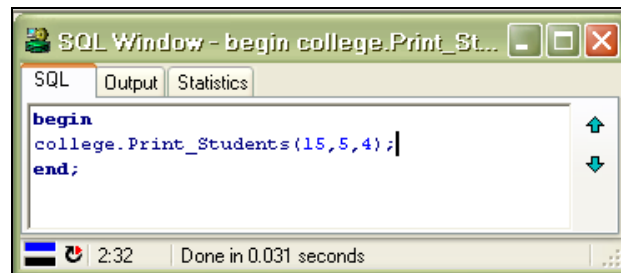
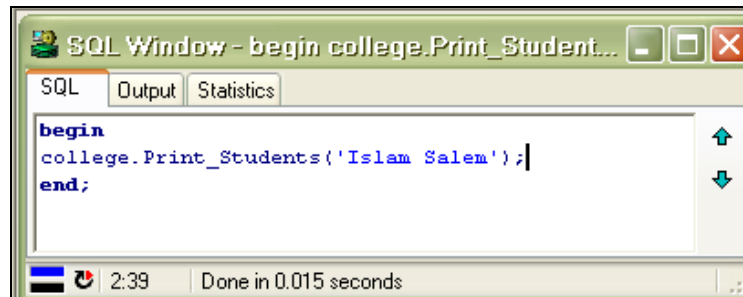
SQL Window - select \* from teachers;

```
select * from teachers;
```

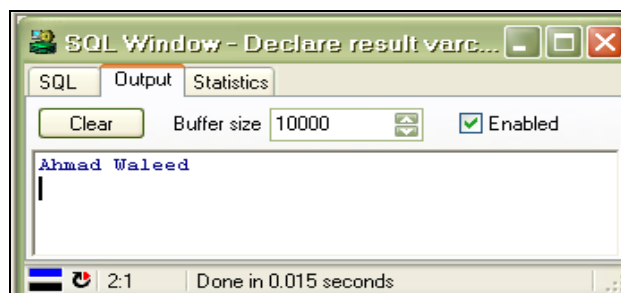
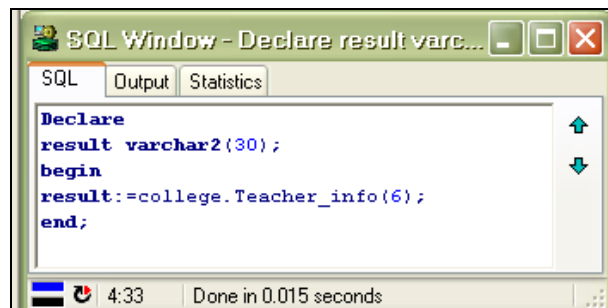
	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	6
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	15	Ayah	Ahmed	6
8	90	Ahmed	ed A	6

1:24 8 rows selected in 0.031 seconds

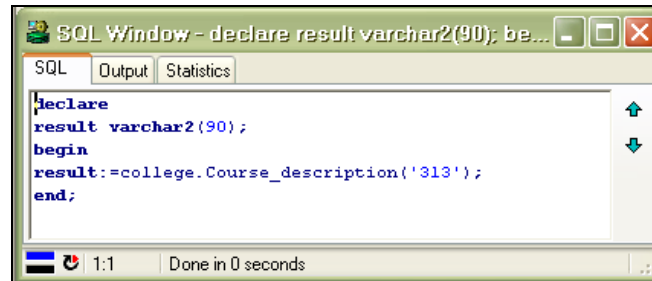
## 11. Print\_Students



## 12. Teacher\_info



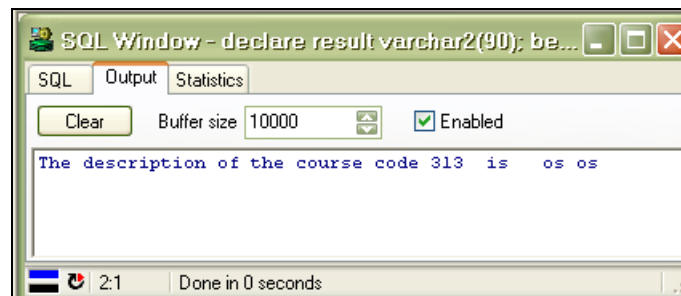
### 13. Course\_description



The screenshot shows an SQL Window titled "SQL Window - declare result varchar2(90); be...". It has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying a PL/SQL block:

```
declare  
result varchar2(90);  
begin  
result:=college.Course_description('313');  
end;
```

At the bottom, there is a status bar showing a zoom level of 1:1 and the text "Done in 0 seconds".



The screenshot shows the same SQL Window, but now the "Output" tab is active. It displays the result of the PL/SQL block execution:

```
The description of the course code 313 is os os
```

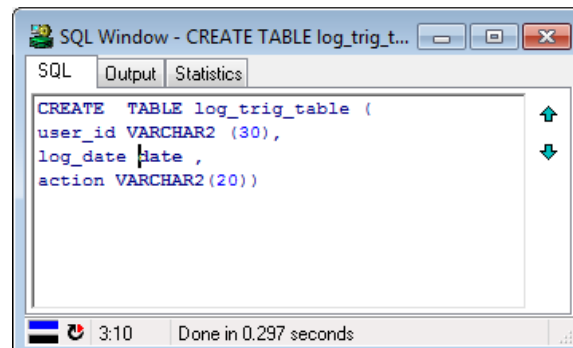
At the bottom, the status bar shows a zoom level of 2:1 and the text "Done in 0 seconds".

## Triggers Part:

### Logon and logoff triggers

Create **logon\_trigger** that add "user\_id, log\_date and action('Logging on')" on log\_trig\_table

- First, we have to create log\_trig\_table

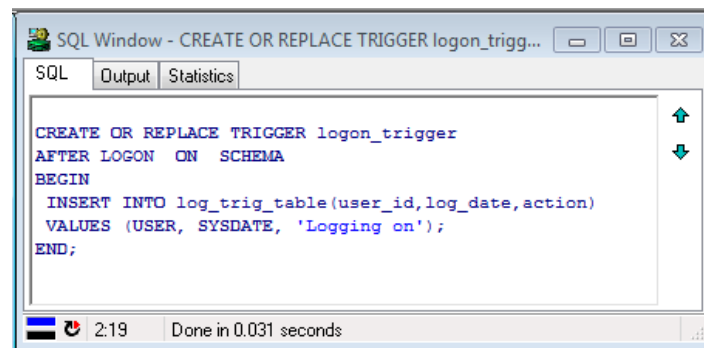


The screenshot shows a SQL Window titled "SQL Window - CREATE TABLE log\_trig\_t...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following SQL code:

```
CREATE TABLE log_trig_table (  
  user_id VARCHAR2 (30),  
  log_date date ,  
  action VARCHAR2(20))
```

At the bottom of the window, there is a status bar showing a progress indicator, the time "3:10", and the message "Done in 0.297 seconds".

- logon\_trigger**

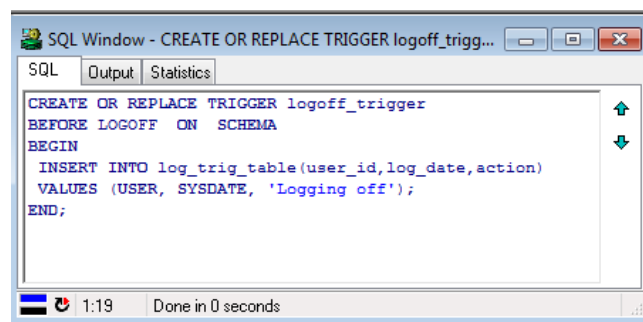


The screenshot shows a SQL Window titled "SQL Window - CREATE OR REPLACE TRIGGER logon\_trigg...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following SQL code:

```
CREATE OR REPLACE TRIGGER logon_trigger  
AFTER LOGON ON SCHEMA  
BEGIN  
  INSERT INTO log_trig_table(user_id,log_date,action)  
  VALUES (USER, SYSDATE, 'Logging on');  
END;
```

At the bottom of the window, there is a status bar showing a progress indicator, the time "2:19", and the message "Done in 0.031 seconds".

Create **logoff\_trigger** that add "user\_id, log\_date and action('Logging off' )" on log\_trig\_table .



The screenshot shows a SQL Window titled "SQL Window - CREATE OR REPLACE TRIGGER logoff\_trigg...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following SQL code:

```
CREATE OR REPLACE TRIGGER logoff_trigger  
BEFORE LOGOFF ON SCHEMA  
BEGIN  
  INSERT INTO log_trig_table(user_id,log_date,action)  
  VALUES (USER, SYSDATE, 'Logging off');  
END;
```

At the bottom of the window, there is a status bar showing a progress indicator, the time "1:19", and the message "Done in 0 seconds".

- The Result of the two previous triggers

SQL Window - SELECT \* FROM log\_trig\_table;

SQL Output Statistics

```
SELECT *
FROM log_trig_table;
```

	USER_ID	LOG_DATE	ACTION
1	COLLEGE_ADMIN ...	1/12/2011 8:37:07 PM	Logging on ...
2	COLLEGE_ADMIN ...	1/12/2011 8:37:09 PM	Logging off ...
3	COLLEGE_ADMIN ...	1/12/2011 8:44:12 PM	Logging on ...
4	COLLEGE_ADMIN ...	1/12/2011 8:44:25 PM	Logging off ...
5	COLLEGE_ADMIN ...	1/14/2011 1:29:14 PM	Logging on ...
6	COLLEGE_ADMIN ...	1/14/2011 1:31:52 PM	Logging on ...
7	COLLEGE_ADMIN ...	1/14/2011 1:41:05 PM	Logging off ...
8	COLLEGE_ADMIN ...	1/14/2011 1:41:10 PM	Logging off ...
9	COLLEGE_ADMIN ...	1/14/2011 12:29:44 PM	Logging on ...
10	COLLEGE_ADMIN ...	1/14/2011 12:31:09 PM	Logging off ...

## Foreign Keys Triggers

- 1- Head\_Assisstant\_ID in department\_heads table is a foreign key for ID in the teacher table.

Program Window - Edit source of trigger ASSIS\_ID\_FK\_TRG@ORCL2

assis\_id\_fk\_trg

Block Code section Insert

```
CREATE OR REPLACE TRIGGER assis_id_fk_trg
AFTER UPDATE OF Head_Assisst_ID ON departments_heads
FOR EACH ROW
BEGIN
    INSERT INTO Teachers VALUES (:new.Head_Assisst_ID, 'first', 'last', NULL);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        NULL;
END;
```

& 5:52

To test the trigger we will update the assistant id to 10 which is not exists in the teachers table, so the trigger will be fired.

Here the teachers table after firing the trigger

SQL Window - SELECT \* FROM Depa

```
UPDATE departments_heads
SET HEAD_Assist_ID = 10
WHERE HEAD_Assist_ID = 4;
```

	DEPARTMENT	HEAD_ID
1	Compute	6

SQL Window - SELECT \* FROM TEACHERS

```
SELECT * FROM TEACHERS
```

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	...
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	10	first	last	...

Here we will again update the assistant id to 4 which is in the teachers id so the trigger do nothing (NULL)

SQL Window - UPDATE departments...

```
UPDATE departments_heads
SET HEAD_Assist_ID = 4
WHERE HEAD_Assist_ID = 10;
```

3:24 0 rows updated in 0 seconds

No change on the teachers table

SQL Window - SELECT \* FROM TEACHERS

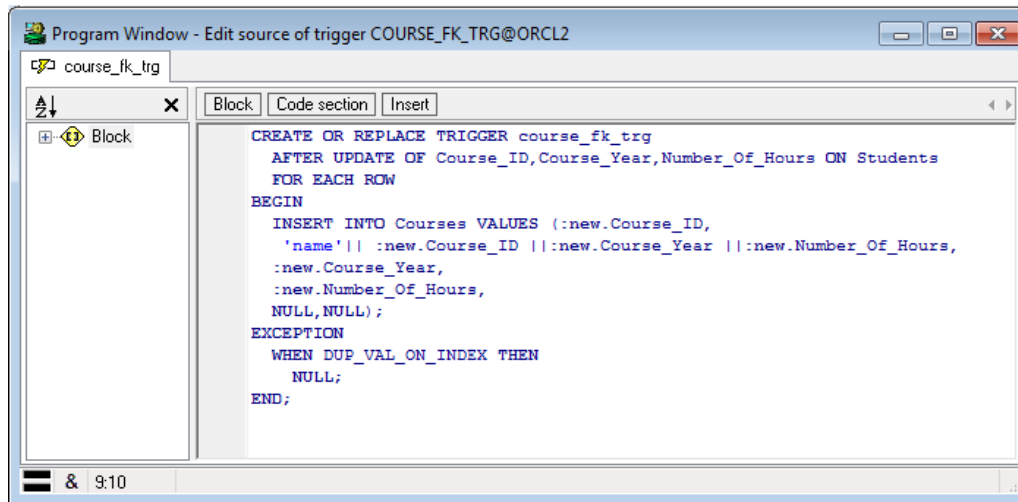
```
SELECT * FROM
TEACHERS
```

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	...
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	10	first	last	...

2:9 7 rows selected in 0.047 seconds

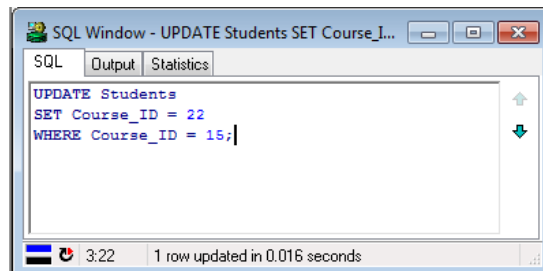


- 2- Trigger for Course\_ID,Year\_ID, Number\_Of\_Hours in Students table which are foreign keys for the same columns in Courses table.



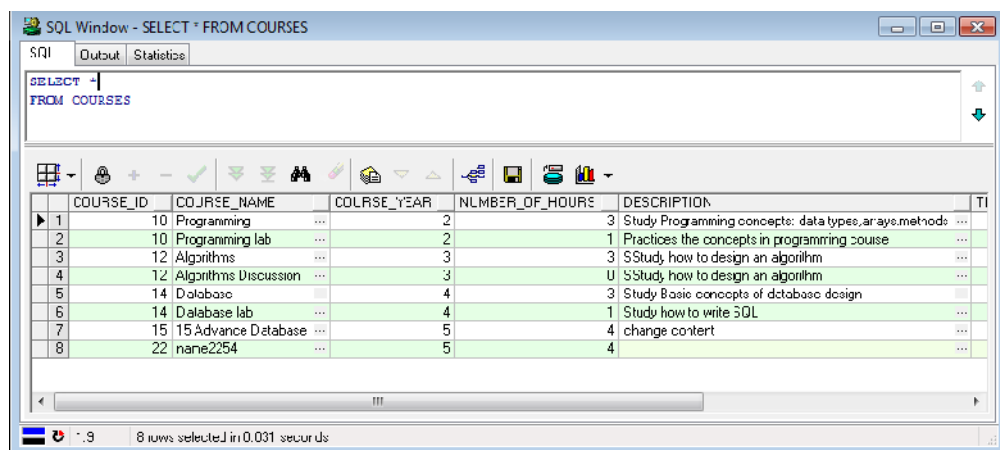
```
CREATE OR REPLACE TRIGGER course_fk_trg
AFTER UPDATE OF Course_ID,Course_Year,Number_Of_Hours ON Students
FOR EACH ROW
BEGIN
INSERT INTO Courses VALUES (:new.Course_ID,
'Name'|| :new.Course_ID ||:new.Course_Year ||:new.Number_Of_Hours,
:new.Course_Year,
:new.Number_Of_Hours,
NULL,NULL);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
NULL;
END;
```

Here the query that will fire the trigger. We update the Course id in the Students table to 22 which is not in the Courses table



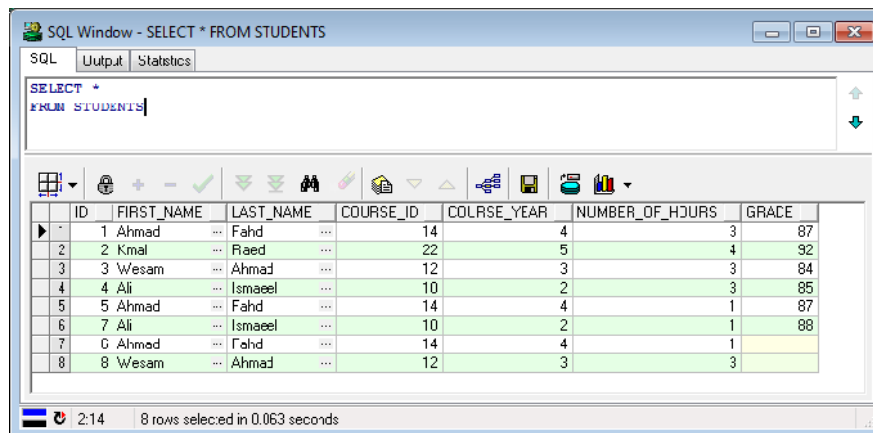
```
UPDATE Students
SET Course_ID = 22
WHERE Course_ID = 15;
```

Here we can see that 22 was added to the courses table



	COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TI
1	10	Programming	...	2	3 Study Programming concepts: data types, arrays, methods	...
2	10	Programming lab	...	2	1 Practices the concepts in programming course	...
3	12	Algorithms	...	3	3 SStudy, how to design an algorithm	...
4	12	Algorithms Discussion	...	3	U SStudy, how to design an algorithm	...
5	14	Database	...	4	3 Study Basic concepts of database design	...
6	14	Database lab	...	4	1 Study how to write SQL	...
7	15	15 Advance Database	...	5	4 change content	...
8	22	name2254	...	5	4	...

Here the Update in the Students table



SQL Window - SELECT \* FROM STUDENTS

SQL Output Statistics

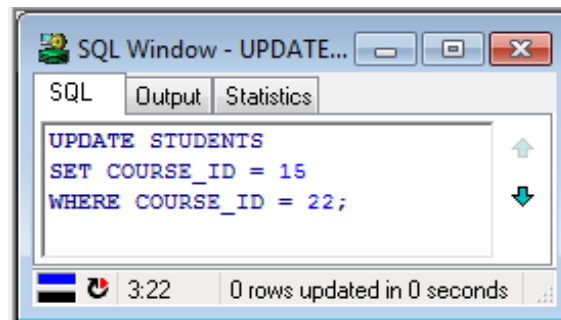
```
SELECT *  
FROM STUDENTS
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	22	5	4	92
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88
7	6	Ahmad	Fahd	14	4	1	
8	8	Wesam	Ahmad	12	3	3	

2:14 8 rows selected in 0.063 seconds

The following query will make the trigger to do nothing (NULL)

The update will not make any problem because the 15 is already in the Courses table



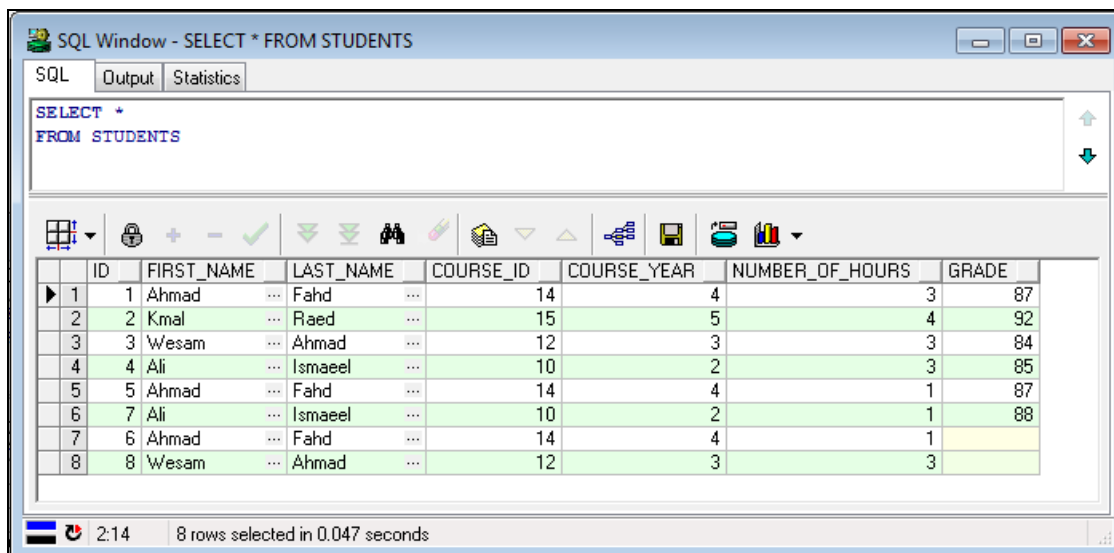
SQL Window - UPDATE...

SQL Output Statistics

```
UPDATE STUDENTS  
SET COURSE_ID = 15  
WHERE COURSE_ID = 22;
```

3:22 0 rows updated in 0 seconds

Here the result of the update



SQL Window - SELECT \* FROM STUDENTS

SQL Output Statistics

```
SELECT *  
FROM STUDENTS
```

	ID	FIRST_NAME	LAST_NAME	COURSE_ID	COURSE_YEAR	NUMBER_OF_HOURS	GRADE
1	1	Ahmad	Fahd	14	4	3	87
2	2	Kmal	Raed	15	5	4	92
3	3	Wesam	Ahmad	12	3	3	84
4	4	Ali	Ismaeel	10	2	3	85
5	5	Ahmad	Fahd	14	4	1	87
6	7	Ali	Ismaeel	10	2	1	88
7	6	Ahmad	Fahd	14	4	1	
8	8	Wesam	Ahmad	12	3	3	

2:14 8 rows selected in 0.047 seconds

- 3- Trigger for the Teacher\_ID in the Courses Table which references to the ID in the teachers table.

```
CREATE OR REPLACE TRIGGER Tech_cou_fk_trg
  AFTER UPDATE OF Teacher_ID ON Courses
  FOR EACH ROW
BEGIN
  INSERT INTO Teachers VALUES (:new.Teacher_ID, 'firsrt', 'last', NULL);
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    NULL;
END;
```

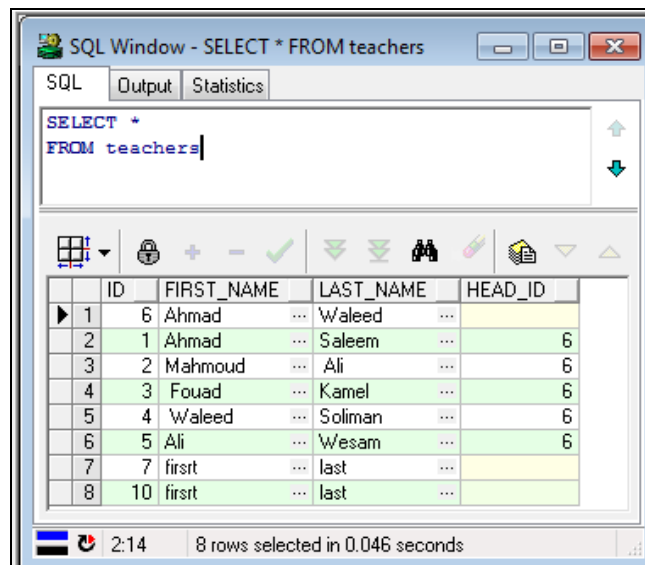
Here the teachers table before firing the trigger

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	10	firsrt	last	

Here the query that will fire the trigger

```
UPDATE COURSES
SET TEACHER_ID = 7
WHERE TEACHER_ID = 1;
```

Here the teachers table after the trigger, we see tha ID = 7 was added to the table



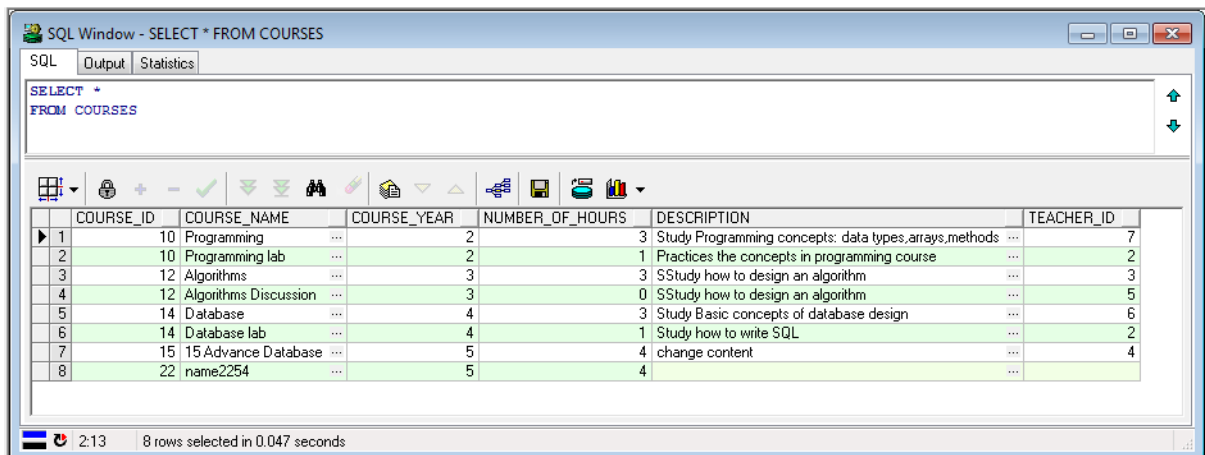
SQL Window - SELECT \* FROM teachers

```
SELECT *
FROM teachers
```

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	7	first	last	
8	10	first	last	

2:14 8 rows selected in 0.046 seconds

Here the update in the courses table



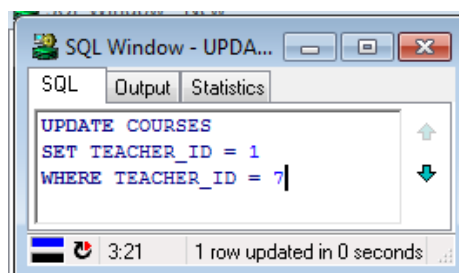
SQL Window - SELECT \* FROM COURSES

```
SELECT *
FROM COURSES
```

	COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	10	Programming	2	3	Study Programming concepts: data types, arrays, methods	7
2	10	Programming lab	2	1	Practices the concepts in programming course	2
3	12	Algorithms	3	3	Study how to design an algorithm	3
4	12	Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14	Database	4	3	Study Basic concepts of database design	6
6	14	Database lab	4	1	Study how to write SQL	2
7	15	Advance Database	5	4	change content	4
8	22	name2254	5	4		

2:13 8 rows selected in 0.047 seconds

Here the update will cause nothing in the trigger and the update will be done without any change in the teachers table.



SQL Window - UPDA...

```
UPDATE COURSES
SET TEACHER_ID = 1
WHERE TEACHER_ID = 7
```

3:21 1 row updated in 0 seconds

Here the courses table after update

	COURSE_ID	COURSE_NAME	COURSE_YEAR	NUMBER_OF_HOURS	DESCRIPTION	TEACHER_ID
1	10	Programming	2	3	Study Programming concepts: data types, arrays, methods	1
2	10	Programming lab	2	1	Practice the concepts in programming course	2
3	12	Algorithms	3	3	Study how to design an algorithm	3
4	12	Algorithms Discussion	3	0	Study how to design an algorithm	5
5	14	Database	4	3	Study Basic concepts of database design	6
6	14	Database lab	4	1	Study how to write SQL	2
7	15	Advance Database	5	4	change content	4
8	22	name2254	5	4		

- 4- Triggers for the foreign key Head\_ID in the departments\_head table which references to the Head\_ID in the teachers table

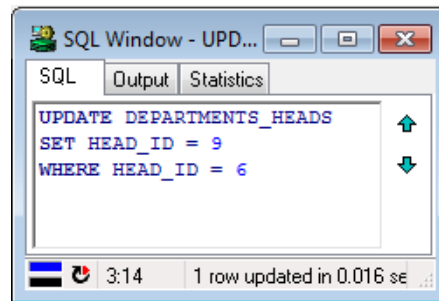
```

CREATE OR REPLACE TRIGGER head_id_dep_fk_trg
AFTER UPDATE OF Head_ID ON departments_heads
FOR EACH ROW
BEGIN
    INSERT INTO Teachers VALUES (:new.Head_ID, 'first', 'last', NULL);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        NULL;
END;
  
```

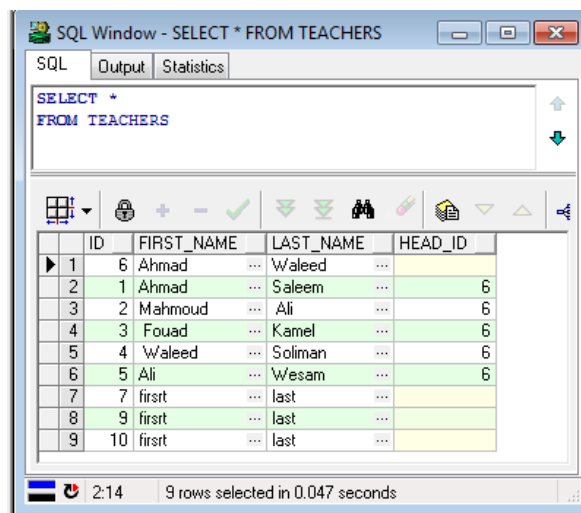
The departments\_head table before the firing of the trigger

	DEPARTMENT	HEAD_ID	HEAD
1	Compute	6	4

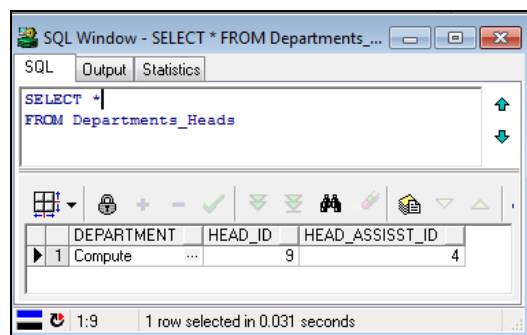
Now we will update the head\_id to 9 which is not in the teachers table which cause firing the trigger that will add the teacher with id = 9 to the teachers table



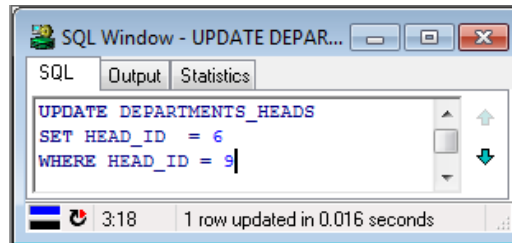
The result of firing the trigger on the teachers table (insertion of teacher with id = 9)



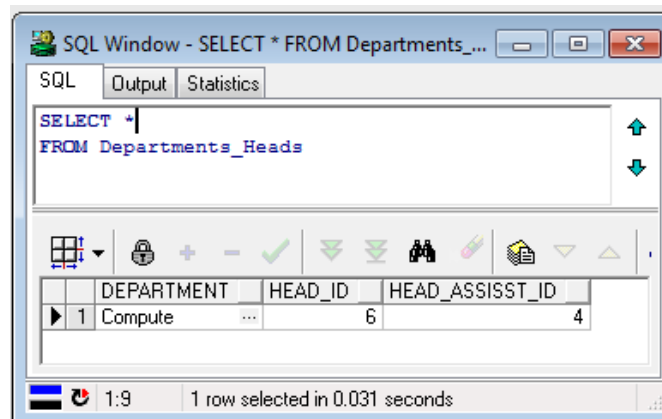
Here the result of updating the departments\_head table



If we update the head\_id with a value that already in the teachers table the trigger will make nothing in the teachers (NULL action)

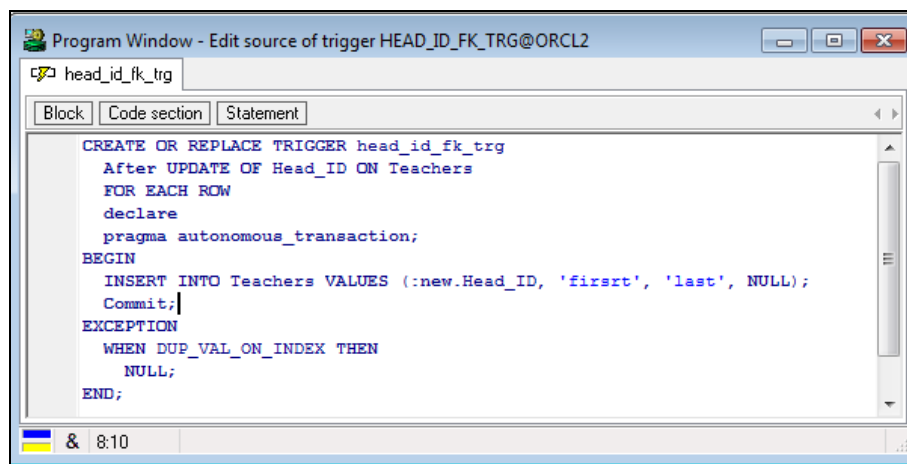


The update in the departments\_head table



- 5- Trigger for the Head\_id in the teachers table which is foreign key for the ID in the same table.

Here we will face a mutating problem which can be solved as the following



Here a brief explanation

Pragma Autonomous transaction:-

Whenever we commit all the transactions before the last commit in that session get saved. Suppose that you have a scenario you just want a particular transaction only to be saved. In order to achieve that you make this as a separate transaction which can be committed but prior to this the transactions are not commit.

eg:

insert into tab1 () values ();

update tab2 set values.. some more stmts..

pragma autonomous transaction

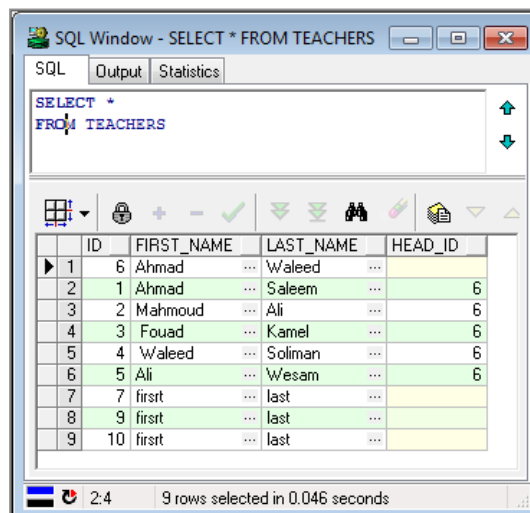
insert into tab3 values();

commit;

here.. insert into tab1 is only saved but not tab1 and tab2.

-----

Here the teachers table before the trigger

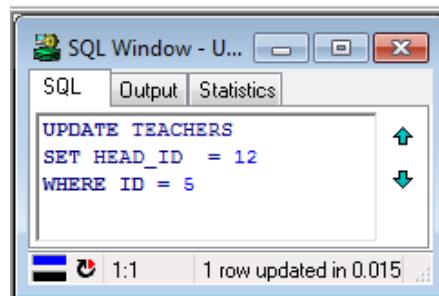


The screenshot shows an SQL Window titled "SQL Window - SELECT \* FROM TEACHERS". The SQL tab is active, displaying the query "SELECT \* FROM TEACHERS". Below the query, there is a table with 9 rows and 5 columns: ID, FIRST\_NAME, LAST\_NAME, and HEAD\_ID. The table data is as follows:

	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	...
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	6
7	7	first	last	...
8	9	first	last	...
9	10	first	last	...

The status bar at the bottom indicates "2:4 9 rows selected in 0.046 seconds".

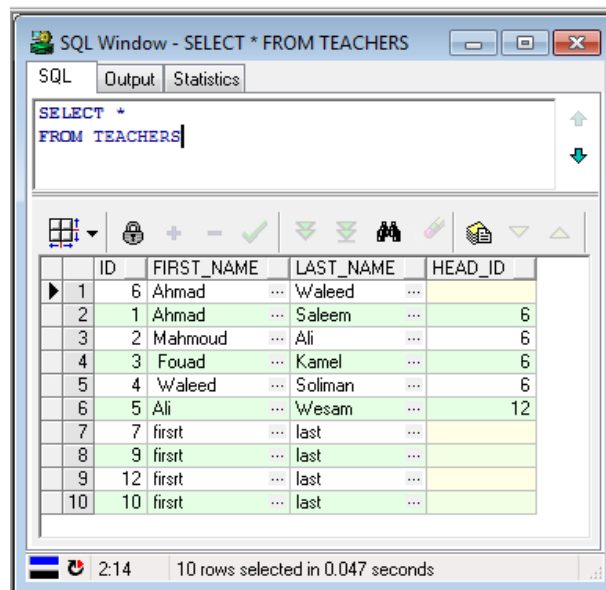
Here the query that will fire the trigger because the id = 12 is not in the teachers table



The screenshot shows an SQL Window titled "SQL Window - U...". The SQL tab is active, displaying the query "UPDATE TEACHERS SET HEAD\_ID = 12 WHERE ID = 5". The status bar at the bottom indicates "1:1 1 row updated in 0.015".



Here the table after firing the table and inserting the teacher with id = 12

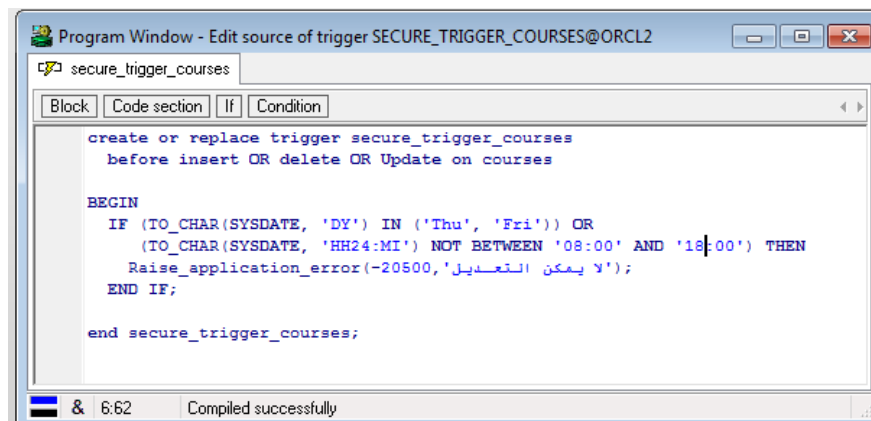


	ID	FIRST_NAME	LAST_NAME	HEAD_ID
1	6	Ahmad	Waleed	...
2	1	Ahmad	Saleem	6
3	2	Mahmoud	Ali	6
4	3	Fouad	Kamel	6
5	4	Waleed	Soliman	6
6	5	Ali	Wesam	12
7	7	first	last	...
8	9	first	last	...
9	12	first	last	...
10	10	first	last	...

2:14 10 rows selected in 0.047 seconds

## Security triggers

- 1- Trigger on the Courses table in order not to update or delete or insert on the table after the end of the work times



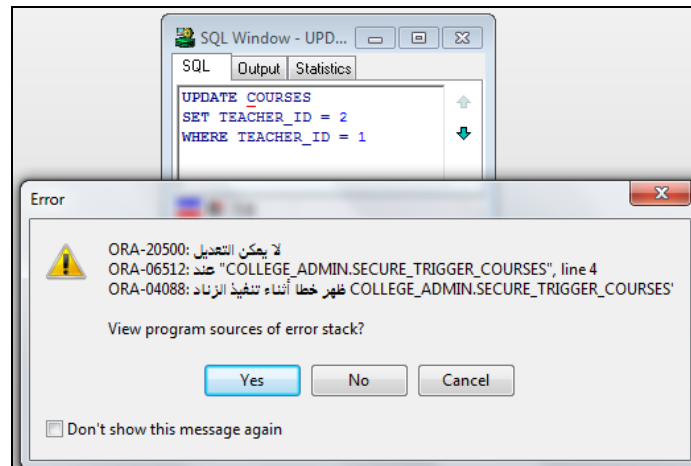
```
create or replace trigger secure_trigger_courses
before insert OR delete OR Update on courses

BEGIN
  IF (TO_CHAR(SYSDATE, 'DY') IN ('Thu', 'Fri')) OR
    (TO_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00') THEN
    Raise_application_error(-20500, 'لا يمكن التعديل');
  END IF;

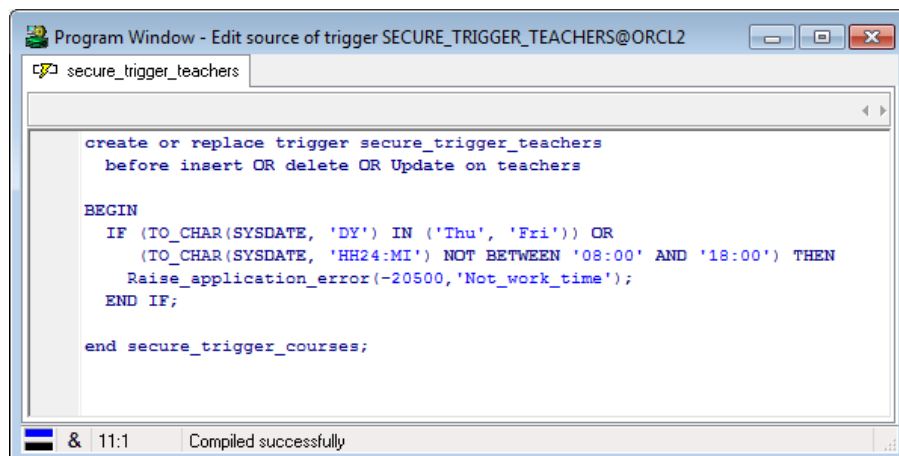
end secure_trigger_courses;
```

& 6:62 Compiled successfully

Here the test of the trigger



- 2- Trigger on the Teachers table in order not to update or delete or insert on the table after the end of the work times



Here the result of the test of this trigger

