

Data Base

Abdalwahab

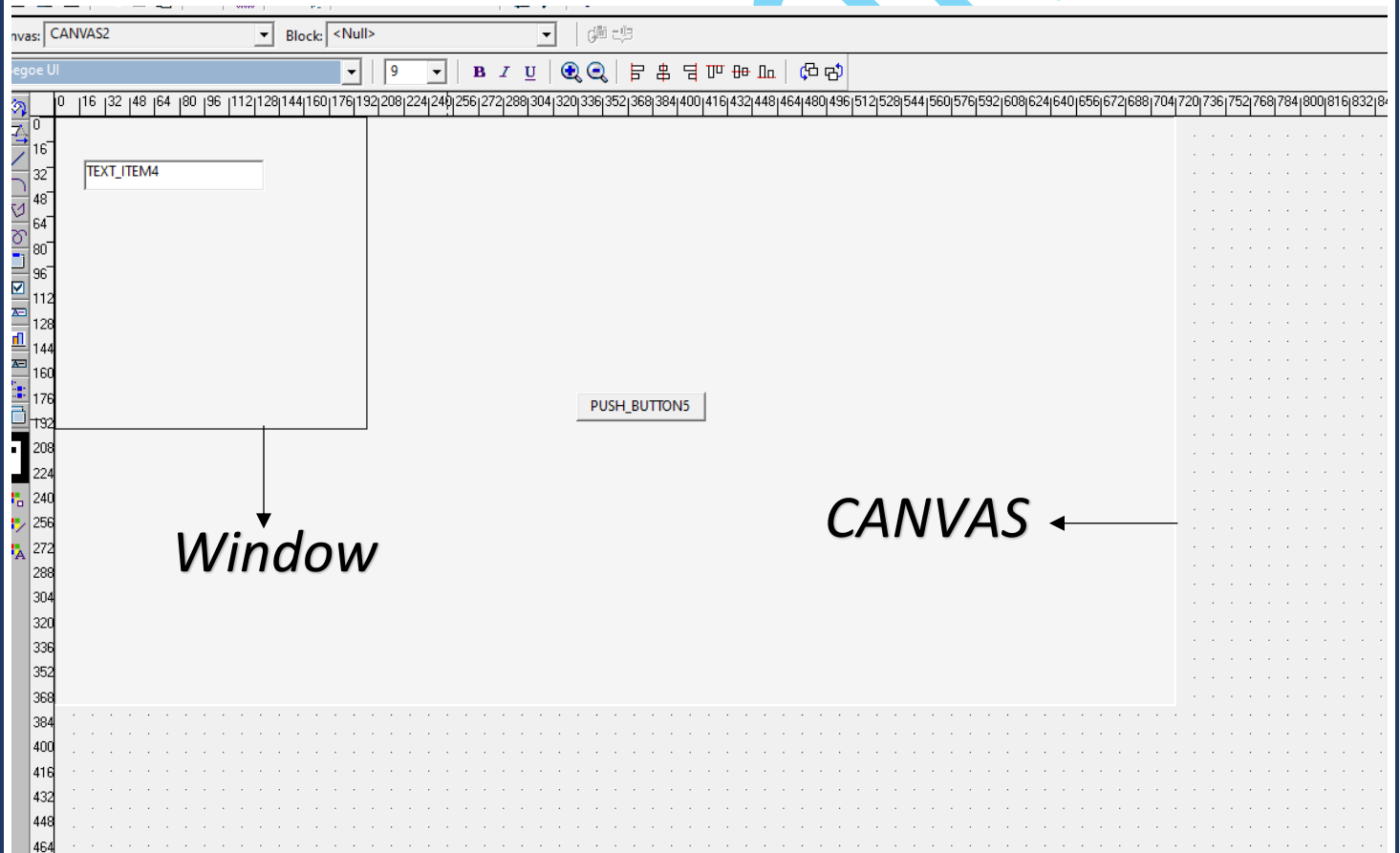
Qatawneh

1. WINDOW

2. ALERT

3. LOV

1. Window:



0 16 32 48 64 80 96 112 128 144 160 176 192 208 224 240 256 272 288 304 320 336 352 368 384 400 416 432 448 464 480 496 512 528 544 560 576 592 608 624 640 656 672 688 704 7

6
12
18
24
30
36
42
48
54
60
66
72
78
84
90
96
102
108
114
120
126
132
138
144
150
156
162
168
174
180
186
192
198
204
210
216
222
228
234
240
246
252
258
264
270
276
282
288
294
300
306
312
318
324
330
336
342
348
354
360
366
372
378
384
390
396
402
408
414
420
426
432
438
444
450
456
462
468
474
480
486
492
498
504
510
516
522
528
534
540
546
552
558
564
570
576
582
588
594
600
606
612
618
624
630
636
642
648
654
660
666
672
678
684
690
696
702
708
714
720
726
732
738
744
750
756
762
768
774
780
786
792
798
804
810
816
822
828
834
840
846
852
858
864
870
876
882
888
894
900
906
912
918
924
930
936
942
948
954
960
966
972
978
984
990
996
1000

TEXT_ITEM4

Forms mdi window

Oracle Developer Forms Runtime - Web

Action Edit Query Block Record Field Help Window

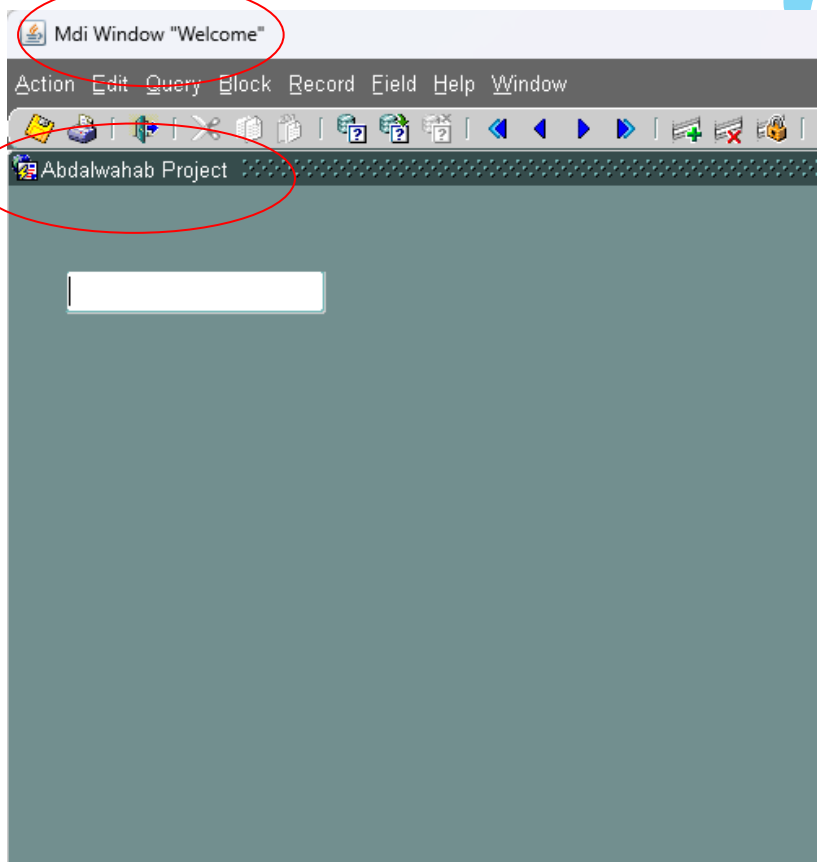
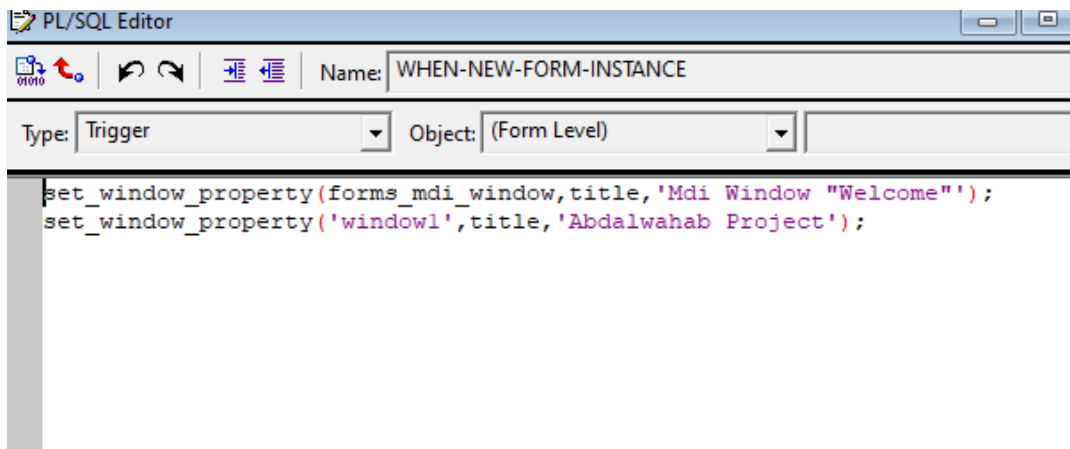
ORACLE

WINDOW1

Window

Record: 1/1

Abd



Oracle Forms Builder - MODULE1 - [PL/SQL Editor]

File Edit View Layout Program Debug Tools Window Help

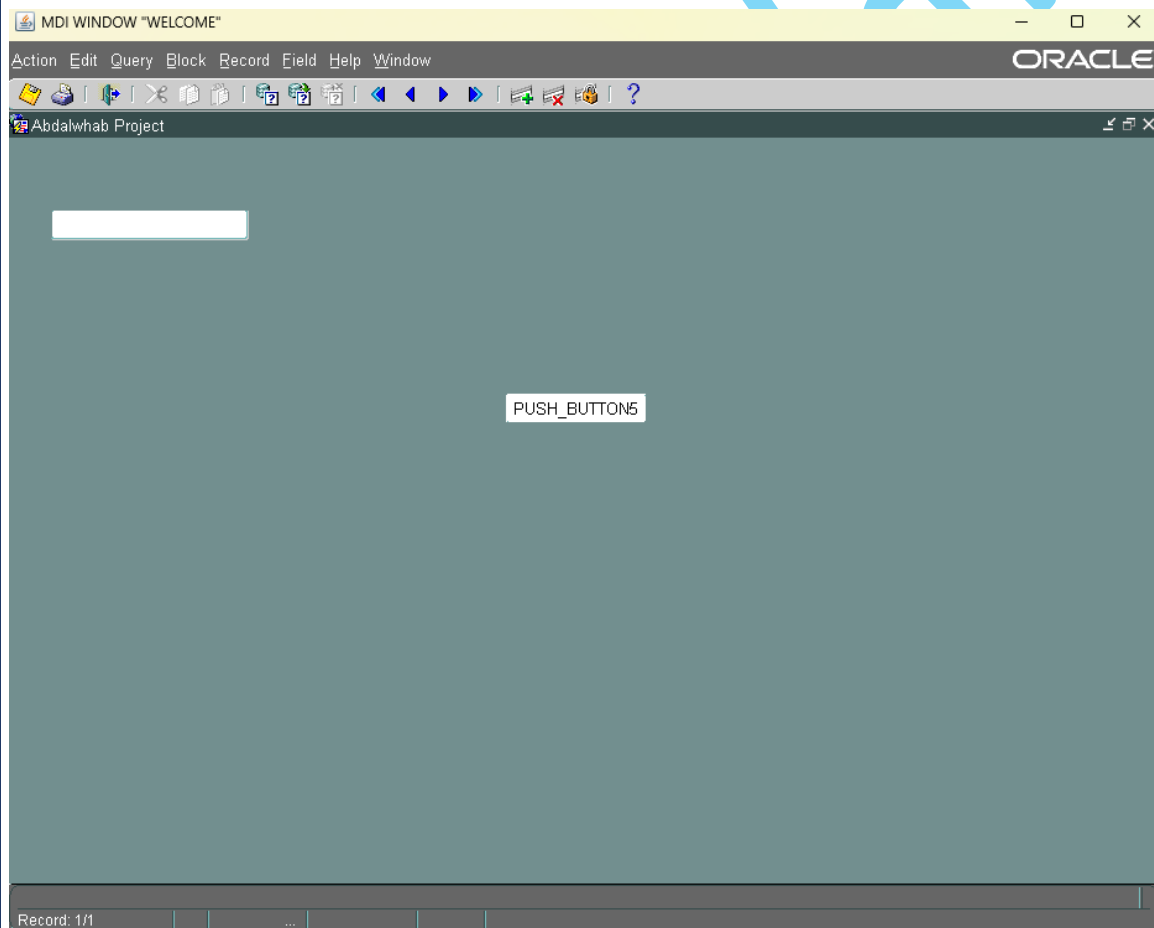
Name: WHEN-NEW-FORM-INSTANCE

Type: Trigger

```
set_window_property(forms_mdi_window,title,'MDI WINDOW "WELCOME"');
set_window_property('window1',title,'Abdalwhab Project');

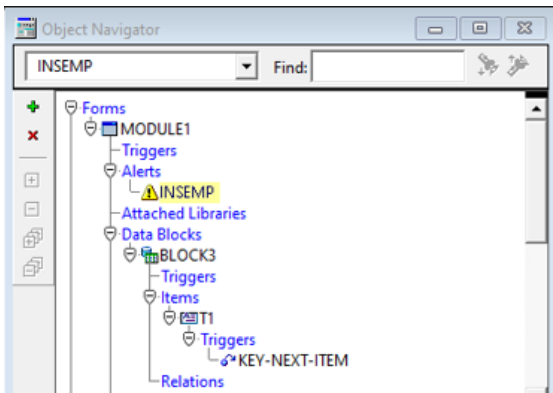
set_window_property(forms_mdi_window>window_state,maximize); → ملء mdi على الشاشة
set_window_property('window1',window_state,maximize); → ملء window على canvas
```

run:

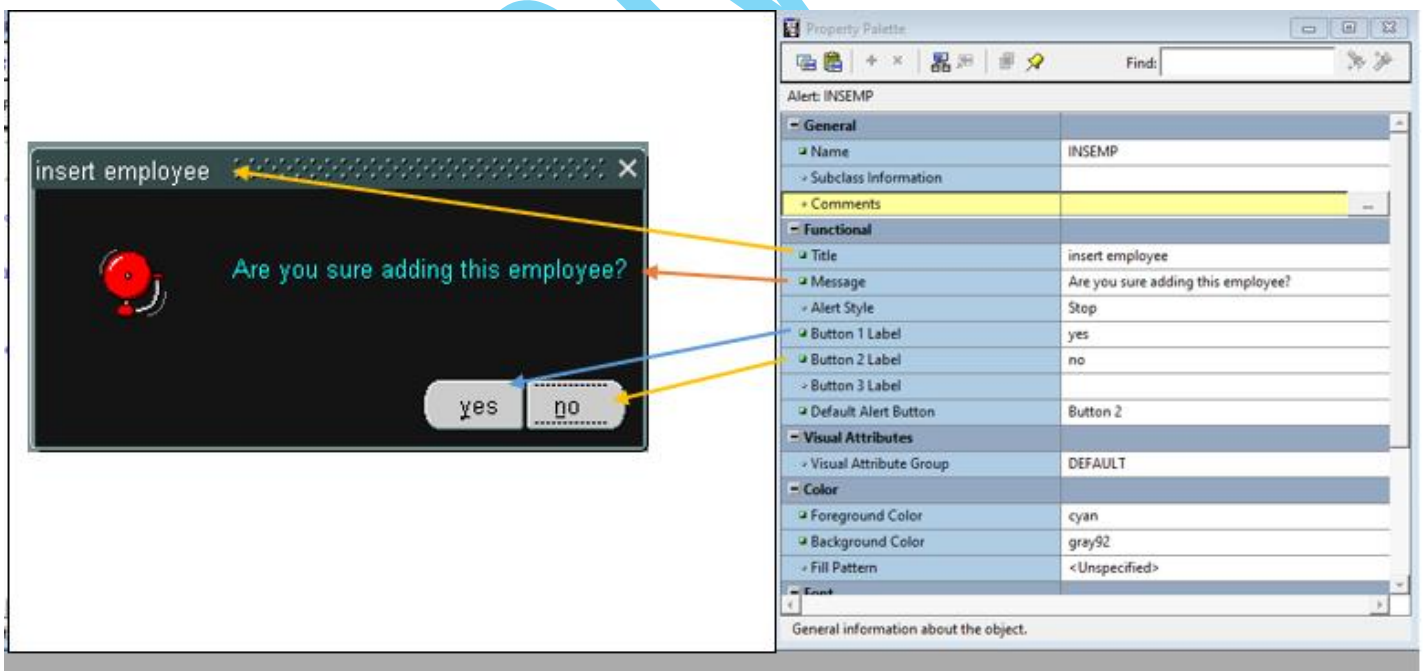


2. Alerts(Dialog Box)

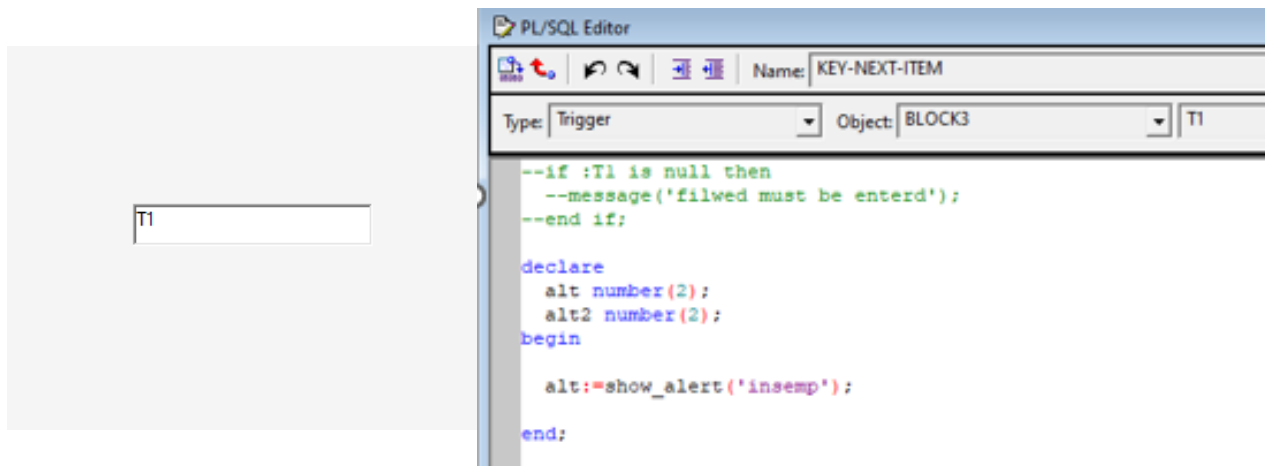
➤ Create Alert



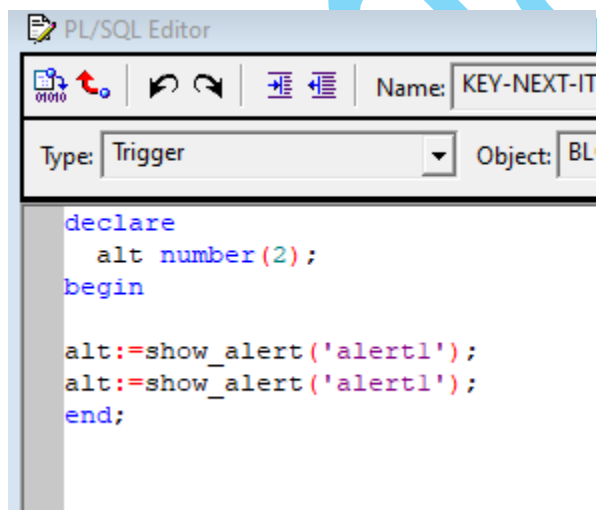
➤ Property palette

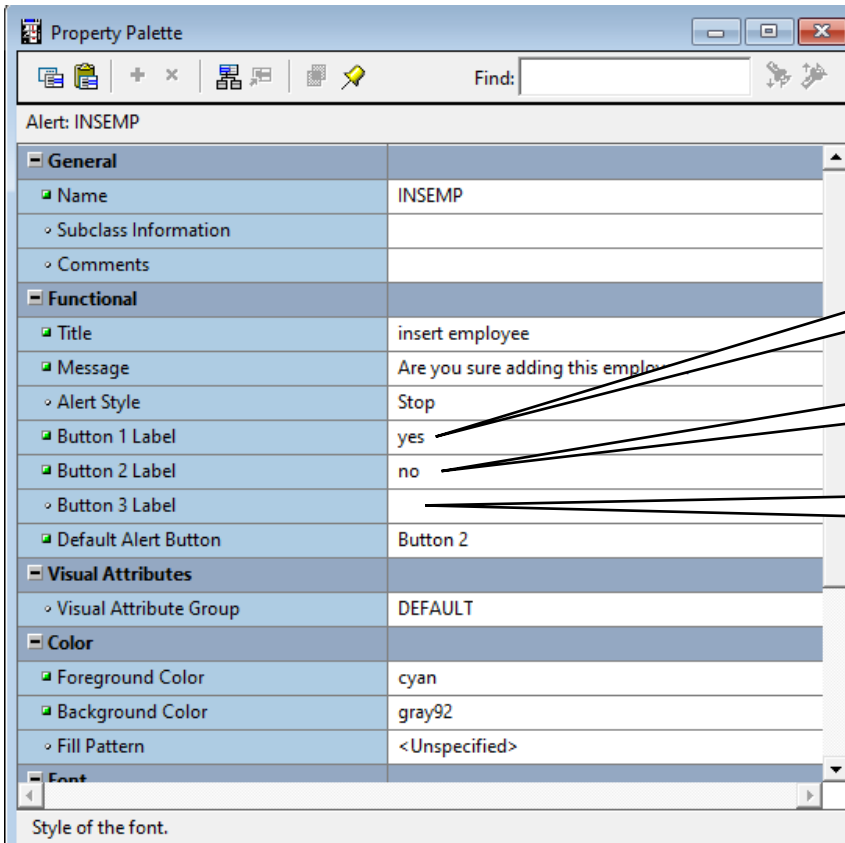


- Call the alert.



- **Show-alert('alert-name').**
- **Returns the value of the button.**
- **values of the button (88,89,90).**
- **Value must be taken.**





Example:

```
set_alert_button_property('alert1', alert_button1, label, 'KKKK');  
set_alert_button_property('alert1', 89, label, 'GGGGG');
```

Alert_button1 = 88

Alert_button2 = 89

Alert_button3 = 90

Example:

The screenshot shows a PL/SQL Editor window with the following details:

- Name:** KEY-NEXT-ITEM
- Type:** Trigger
- Object:** BLOCK3
- Trigger Name:** T1

```
--if :T1 is null then
--message('filwed must be entered');
--end if;

declare
  alt number(2);
  alt2 number(2);
begin

  set_alert_property('insemp',title,'?????');
  set_alert_property('insemp',alert_message_text,'Do you Want to....');

  set_alert_button_property('insemp',alert_button1,label,'OKK');
  set_alert_button_property('insemp',alert_button2,label,'Cancel');

  alt:=show_alert('insemp');
end;
```

A callout box points to the `show_alert` function call, indicating the return values:

- return
- alt = 88
- alt = 89

Block: <Null>

PL/SQL Editor

Name: KEY-NEXT-ITEM

Type: Trigger Object: BLOCK3 T1

```
--if :T1 is null then
--message('filwed must be enterd');
--end if;

declare
  alt number(2);
  alt2 number(2);
begin

  set_alert_property('insemp',title,'??????');
  set_alert_property('insemp',alert_message_text,'Do you Want to.....');

  set_alert_button_property('insemp',88,label,'OKK');
  set_alert_button_property('insemp',89,label,'Cancel');

  alt:=show_alert('insemp');-- 88 / 89

  if alt=88 then
    message('You chose OKK');
  elsif alt=alt2 then
    message('You chose Cancel');
  else
    message('-----');
  end if;

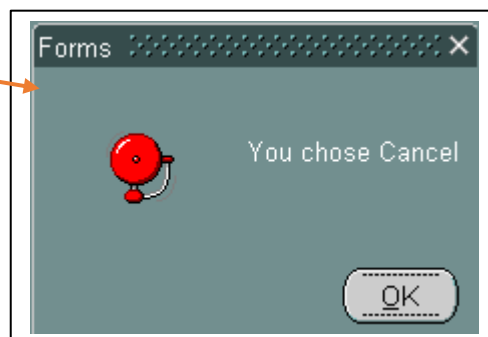
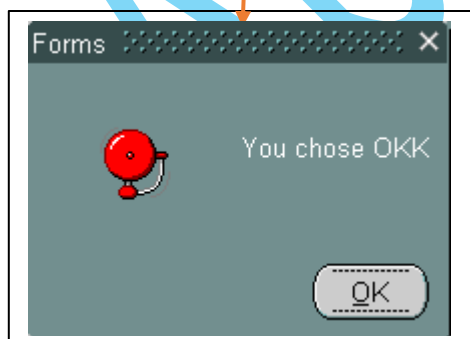
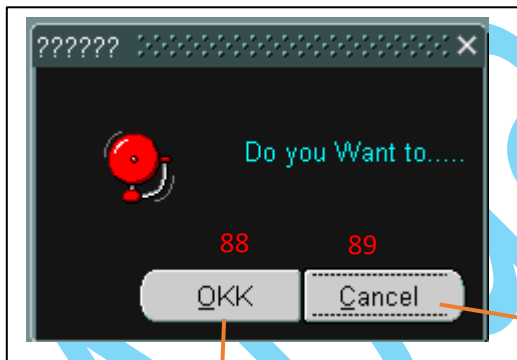
  message(' ');

end;
```

50 365.25


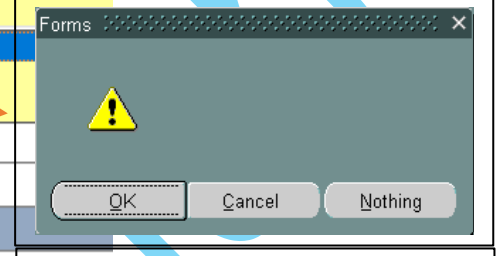

Not Modified

Successfully Compiled



Style Types

General	
Name	ALERT2
Subclass Information	
Comments	
Functional	
Title	
Message	
Alert Style	Stop
Button 1 Label	Stop
Button 2 Label	Caution
Button 3 Label	Note
Button 3 Label	Nothing
Default Alert Button	Button 1
Visual Attributes	
Visual Attribute Group	DEFAULT
Color	
Foreground Color	<Unspecified>
Background Color	<Unspecified>



Abdawn

Ex Use functions & Procedure (alert1, alert2)

The screenshot displays the Oracle Forms Designer interface. On the left, the **Property Palette** for **Alert: ALERT1** is shown. It includes sections for **General**, **Functional**, **Visual Attributes**, **Color**, and **Font**. The **General** section shows the Name as **ALERT1**. The **Functional** section shows the Alert Style as **Stop**, Button 1 Label as **OK**, and Button 2 Label as **Cancel**. The **Visual Attributes** section shows the Visual Attribute Group as **DEFAULT**. The **Color** section shows the Foreground Color, Background Color, and Fill Pattern as **<Unspecified>**. The **Font** section shows the Font Name, Font Size, Font Weight, and Font Style as **<Unspecified>**.

On the right, the **Object Navigator** shows the hierarchy of the form. The **Forms** folder contains **MODULE1**, which includes **Triggers**, **Alerts**, **Attached Libraries**, **Data Blocks**, **Canvases**, **Editors**, **LOVs**, **Object Groups**, **Parameters**, **Popup Menus**, **Program Units**, **Property Classes**, **Record Groups**, **Reports**, **Visual Attributes**, **Windows**, **Menus**, **PL/SQL Libraries**, **Object Libraries**, and **Built-in Packages**. The **Alerts** folder contains **ALERT1** and **ALERT2**. The **Data Blocks** folder contains **BLOCK4**, which includes **Triggers**, **Items**, and **Relations**. The **Items** folder contains **T1**, which includes **Triggers** and **KEY-NEXT-ITEM**.

The screenshot displays the Oracle Forms Designer interface. On the left, the **Property Palette** for **Alert: ALERT2** is shown. It includes sections for **General**, **Functional**, **Visual Attributes**, **Color**, and **Font**. The **General** section shows the Name as **ALERT2**. The **Functional** section shows the Alert Style as **Stop**, Button 1 Label as **OK**, and Button 2 Label as **Cancel**. The **Visual Attributes** section shows the Visual Attribute Group as **DEFAULT**. The **Color** section shows the Foreground Color, Background Color, and Fill Pattern as **<Unspecified>**. The **Font** section shows the Font Name, Font Size, Font Weight, and Font Style as **<Unspecified>**.

On the right, the **Object Navigator** shows the hierarchy of the form. The **Forms** folder contains **MODULE1**, which includes **Triggers**, **Alerts**, **Attached Libraries**, **Data Blocks**, **Canvases**, **Editors**, **LOVs**, **Object Groups**, **Parameters**, **Popup Menus**, **Program Units**, **Property Classes**, **Record Groups**, **Reports**, **Visual Attributes**, **Windows**, **Menus**, **PL/SQL Libraries**, **Object Libraries**, and **Built-in Packages**. The **Alerts** folder contains **ALERT1** and **ALERT2**. The **Data Blocks** folder contains **BLOCK4**, which includes **Triggers**, **Items**, and **Relations**. The **Items** folder contains **T1**, which includes **Triggers** and **KEY-NEXT-ITEM**.

➤ Program units

The screenshot displays the Oracle Developer interface with the PL/SQL Editor and Object Navigator.

PL/SQL Editor:

- Name: FUN (Function Body)
- Type: Program Unit
- Object: (empty)
- Code:

```
FUNCTION fun(v_text varchar2) RETURN boolean IS
BEGIN
  set_alert_property('alert1',alert_message_text,v_text);

  if show_alert('alert1') = 88 then
    return true;
  else
    return false;
  end if;
END;
```
- Status: Not Modified, Successfully Compiled

Object Navigator:

- Selected: FUN (Function Body)
- Tree structure:
 - Forms
 - MODULE1
 - Triggers
 - Alerts
 - ALERT2
 - ALERT1
 - Attached Libraries
 - Data Blocks
 - BLOCK4
 - Triggers
 - Items
 - T1
 - Triggers
 - KEY-NEXT-ITEM
 - Relations
 - Canvases
 - CANVAS3
 - Editors
 - LOVs
 - Object Groups
 - Parameters
 - Popup Menus
 - Program Units
 - FUN (Function Body) (selected)
 - MSG (Procedure Body)
 - Property Classes
 - Record Groups
 - Reports

The screenshot displays the Oracle Developer interface with the PL/SQL Editor and Object Navigator.

PL/SQL Editor:

- Name: MSG (Procedure Body)
- Type: Program Unit
- Object: (empty)
- Code:

```
PROCEDURE msg(v_text varchar2) IS
  alt number(2);
BEGIN
  set_alert_property('alert2',alert_message_text,v_text);
  alt:=show_alert('alert2');
END;
```
- Status: Not Modified, Successfully Compiled

Object Navigator:

- Selected: MSG (Procedure Body)
- Tree structure (identical to the previous screenshot):
 - Forms
 - MODULE1
 - Triggers
 - Alerts
 - ALERT2
 - ALERT1
 - Attached Libraries
 - Data Blocks
 - BLOCK4
 - Triggers
 - Items
 - T1
 - Triggers
 - KEY-NEXT-ITEM
 - Relations
 - Canvases
 - CANVAS3
 - Editors
 - LOVs
 - Object Groups
 - Parameters
 - Popup Menus
 - Program Units
 - FUN (Function Body)
 - MSG (Procedure Body) (selected)
 - Property Classes
 - Record Groups
 - Reports

T1

PL/SQL Editor

Name: KEY-NEXT-ITEM

Type: Trigger Object: BLOCK4 T1

```
declare
  bo boolean;
begin
  bo:=fun('Select Button:');

  if bo then
    msg('OKK');
  else
    msg('Cancel');
  end if;

end;
```

Modified Not Compiled

Forms

Select Button:

OK Cancel

Forms

OKK

OK

Forms

Cancel

OK

Ex: alert message, when entering test and processing the buttons

(Ex: Button name printing).

Alert: TEST	
General	
Name	TEST
Subclass Information	
Comments	
Functional	
Title	
Message	
Alert Style	Caution
Button 1 Label	OK
Button 2 Label	Cancel
Button 3 Label	Nothing
Default Alert Button	Button 1
Visual Attributes	
Visual Attribute Group	DEFAULT

1. First Way

Enter any text
T1
in same trigger

Enter any text
T2
using Procedure

Enter any text
T3
using function

P1

PL/SQL Editor

Name: KEY-NEXT-ITEM

Type: Trigger Object: BLOCK2 T1

```
declare
    alt number(2);
begin
    set_alert_property('test',alert_message_text,:T1);

    alt:=show_alert('test');

    if alt = 88 then
        message('You Chose Yes');
    elsif alt=89 then
        message('You Chose No');
    elsif alt=90 then
        message('You Chose Nothing');
    end if;
end;
```

Not Modified Successfully Compiled

2. Second Way (using Procedure)

using procedure using function

The PL/SQL Editor window shows the following code for the MSG (Procedure Body):

```
PROCEDURE msg(v_text varchar2) IS
alt number(2);
BEGIN
  set_alert_property('test',alert_message_text,v_text);
  alt:=show_alert('test');

  if alt = 88 then
    message('You Chose Yes');
  elsif alt = 89 then
    message('You Chose No');
  elsif alt = 90 then
    message('You Chose Nothing');
  end if;
  message(' ');
END;
```

The Object Navigator window shows the database schema structure. The MSG (Procedure Body) is highlighted under the Triggers folder, indicating its location within the database.

Enter any text

T1

in same trigger

Enter any text

T2

using Procedure

Enter any text

T3

using function

P1

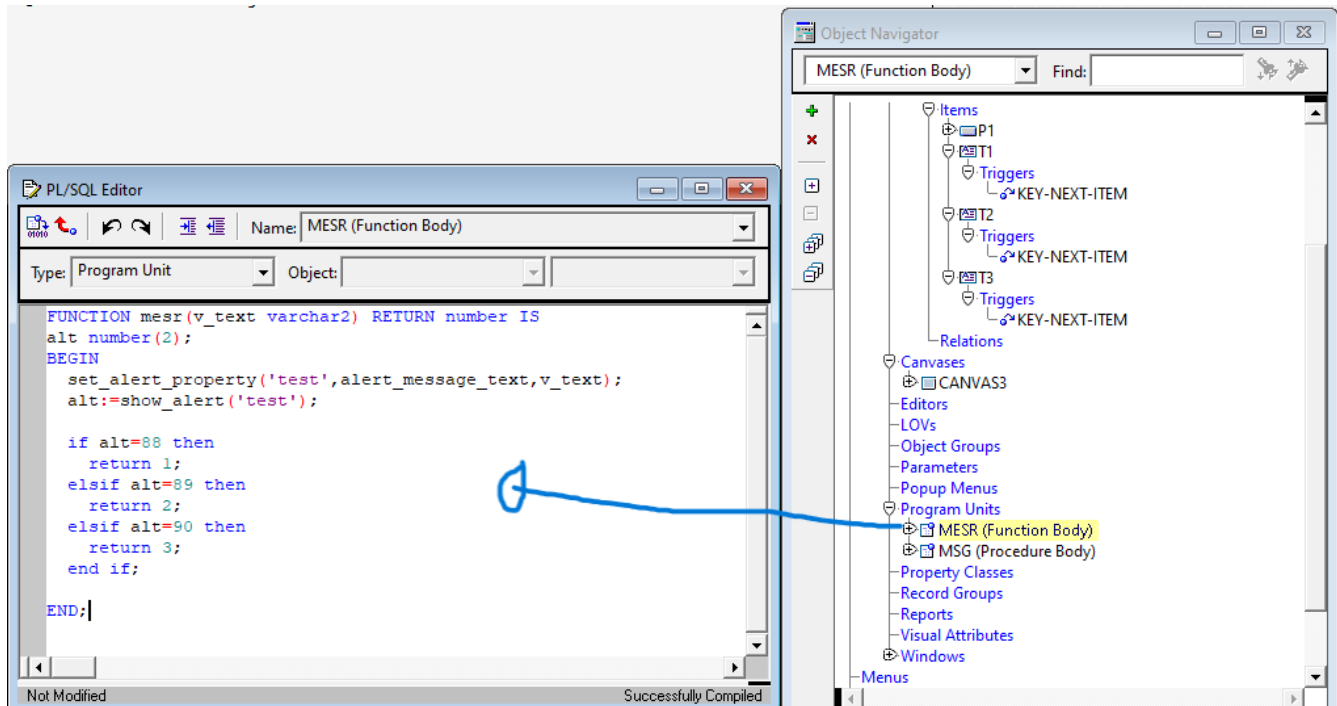
PL/SQL Editor

Name: KEY-NEXT-ITEM

Type: Trigger Object: BLOCK2 T2

msg (:T2);

3. Third Way (using function)



Enter any text

T1

in same trigger

Enter any text

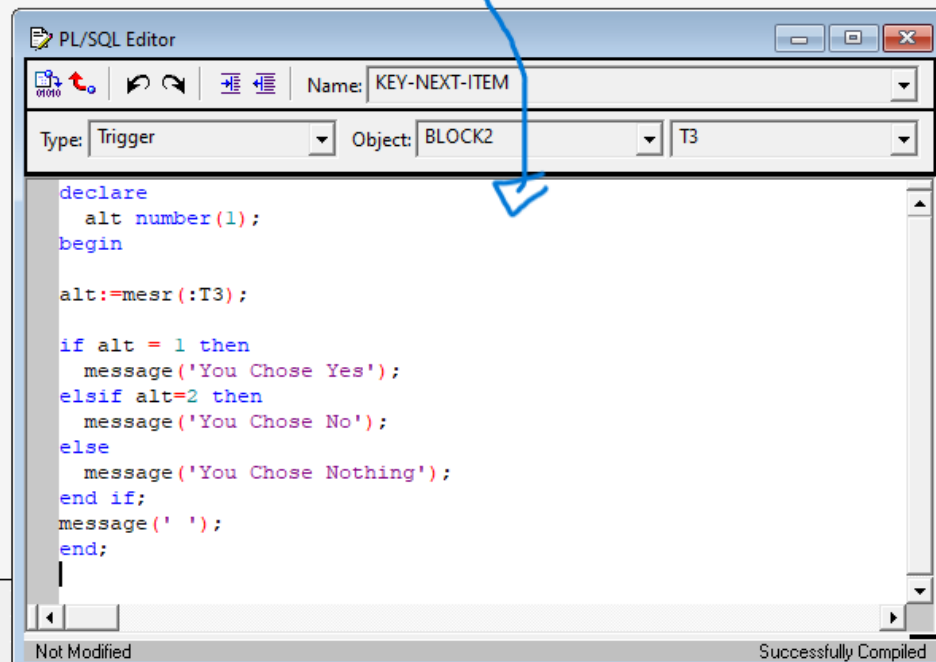
T2

using Procedure

Enter any text

T3

using function



Node:

The alert may not be called more than once.

The screenshot shows the PL/SQL Editor with a function named `MESR` (Function Body). The function code is as follows:

```
FUNCTION mesr(v_text varchar2) RETURN number IS
BEGIN
  set_alert_property('test',alert_message_text,v_text);

  if show_alert('test')=88 then
    return 1;
elseif show_alert('test')=89 then
    return 2;
elseif show_alert('test')=90 then
    return 3;
end if;

END;
```

The code is marked as "Not Modified" and "Successfully Compiled". The Object Navigator on the right shows the hierarchy of the application, with "MESR (Function Body)" selected under "Program Units". A blue arrow points from the "MESR (Function Body)" node in the Object Navigator to the function code in the PL/SQL Editor.

The screenshot shows the PL/SQL Editor with the same function named `MESR` (Function Body). The function code is as follows:

```
FUNCTION mesr(v_text varchar2) RETURN number IS
alt number(2);
BEGIN
  set_alert_property('test',alert_message_text,v_text);

  if show_alert('test')=88 then
    return 1;
  else
    return 2;
  end if;

END;
```

➤ **LOVS (list of vales)** هي قائمة لعرض بيانات من عامود معين ثم اختيار عنصر

Ex



Lovs get their data from **record groups**.

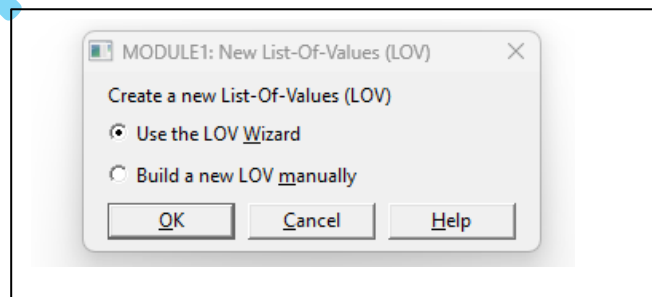
- Record groups: Data Selector.

يسترجع (بيانات/أعمدة) من جدول معين ويمكن إضافة هذه الأعمدة الى

LOVS

Create LOVs:

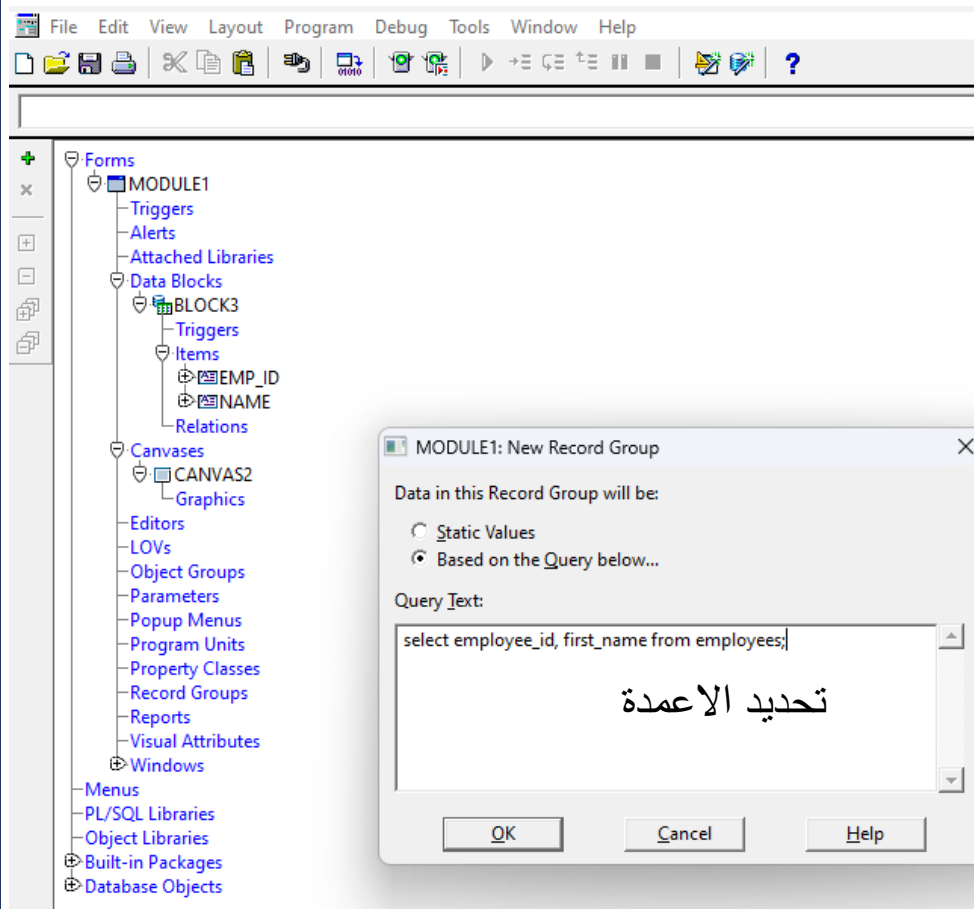
1. Use the LOV wizard.
2. Build new LOV manually.



- Using (build new LOV **manually**)

Steps:

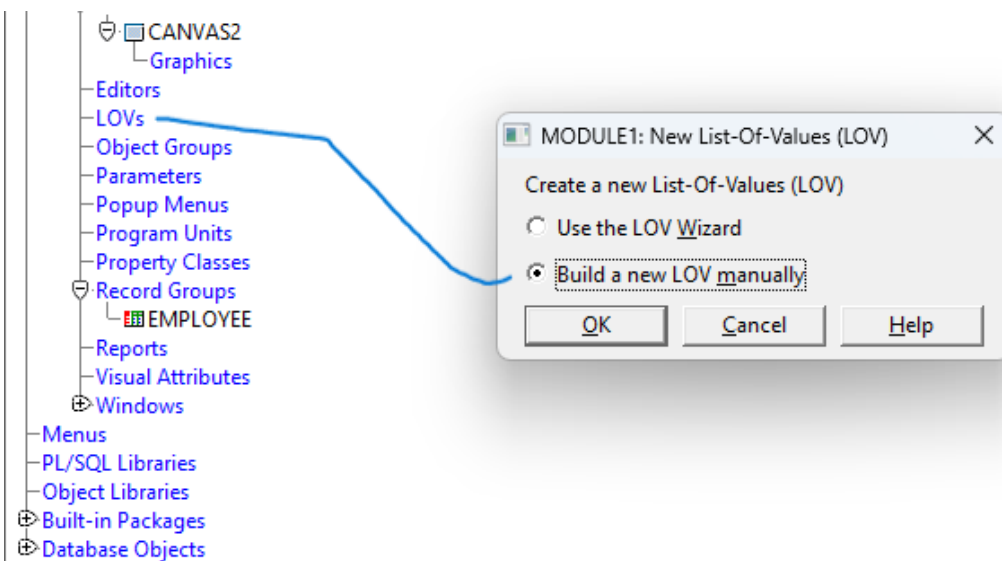
1. A Record Group must be built.



Record Group: EMPLOYEE	
General	
Name	EMPLOYEE تحديد الاسم
Subclass Information	
Comments	
Functional	
Record Group Type	Query
Record Group Query	select employee_id, first_name from employees
Record Group Fetch Size	0
Column Specifications	

يمكن تعديل الجملة الاستعلامية من هنا

2. Create new LOVs.



LOV: EMPLOYEE

General	
Name	EMPLOYEE
Subclass Information	
Comments	
Functional	
Title	Employee id & First Name
List Type	Record Group
Record Group	EMPLOYEE
Column Mapping Properties	More...
Filter Before Display	No
Automatic Display	No
Automatic Refresh	Yes
Automatic Select	No
Automatic Skip	No
Automatic Position	No
Automatic Column Width	No
Physical	

Define Record Group

تحديد الأعمدة المراد عرضها في القائمة

Comments	
Functional	
Title	Employee id & First
List Type	Record Group
Record Group	EMPLOYEE
Column Mapping Properties	
Filter Before Display	No
Automatic Display	No
Automatic Refresh	Yes
Automatic Select	No
Automatic Skip	No
Automatic Position	No
Automatic Column Width	No
Physical	
X Position	
Y Position	
Width	
Height	
Visual Attributes	
Visual Attribute Group	DEFAULT
Color	

LOV Column Mapping

Column Names

EMPLOYEE_ID

Return Item

Display Width

1

Column Title

anything

OK Cancel Help

اسم العمود المراد اضافته

اختيار عنصر يأخذ القيمة التي تم اختارتها من القائمة

LOV Column Mapping

Column Names

FIRST_NAME

EMPLOYEE_ID

Return Item

BLOCK3.NAME

Display Width

90

Column Title

Fisrt Name

OK Cancel Help

LOV Column Mapping

Column Names

FIRST_NAME

EMPLOYEE_ID

Return Item

BLOCK3.EMP_ID

Display Width

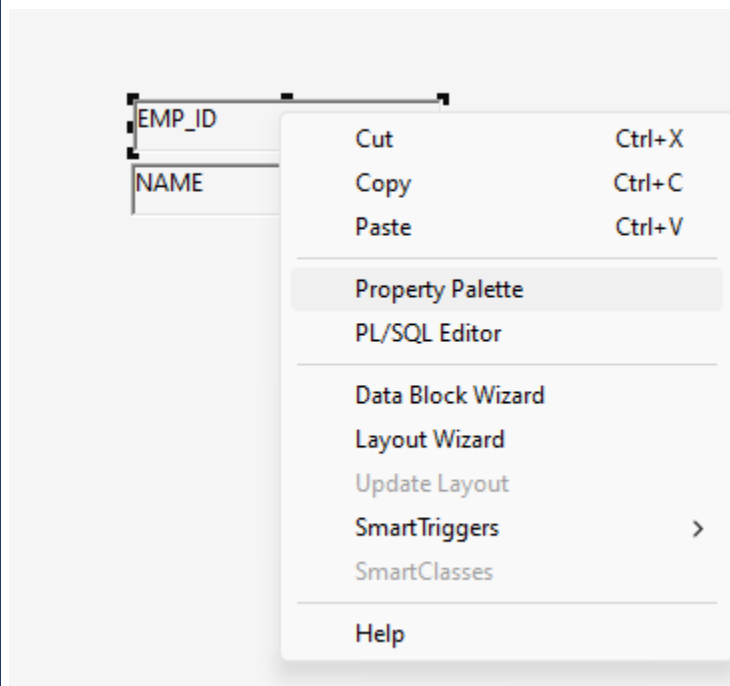
90

Column Title

anything

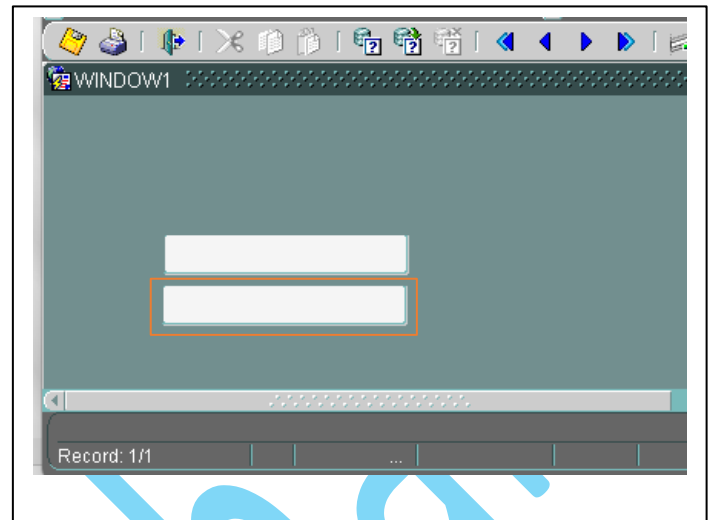
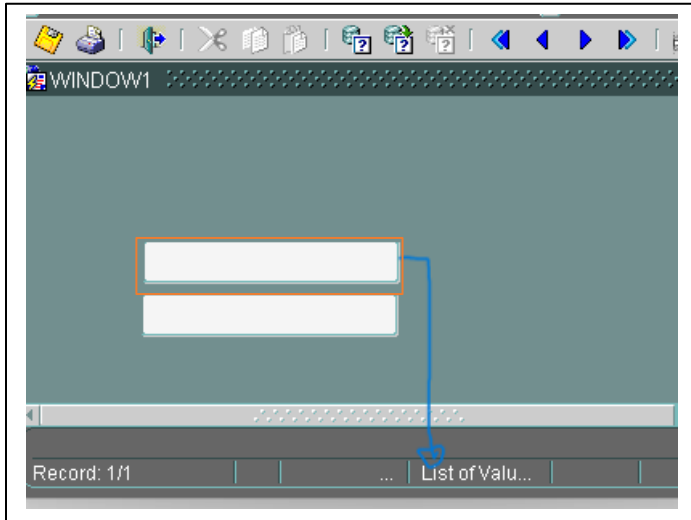
OK Cancel Help

3. Connect item with LOVs. يجب تحديد مكان ظهور القائمة

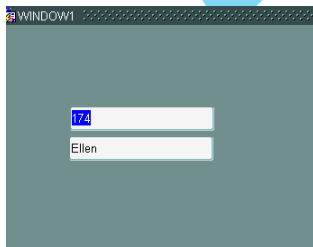
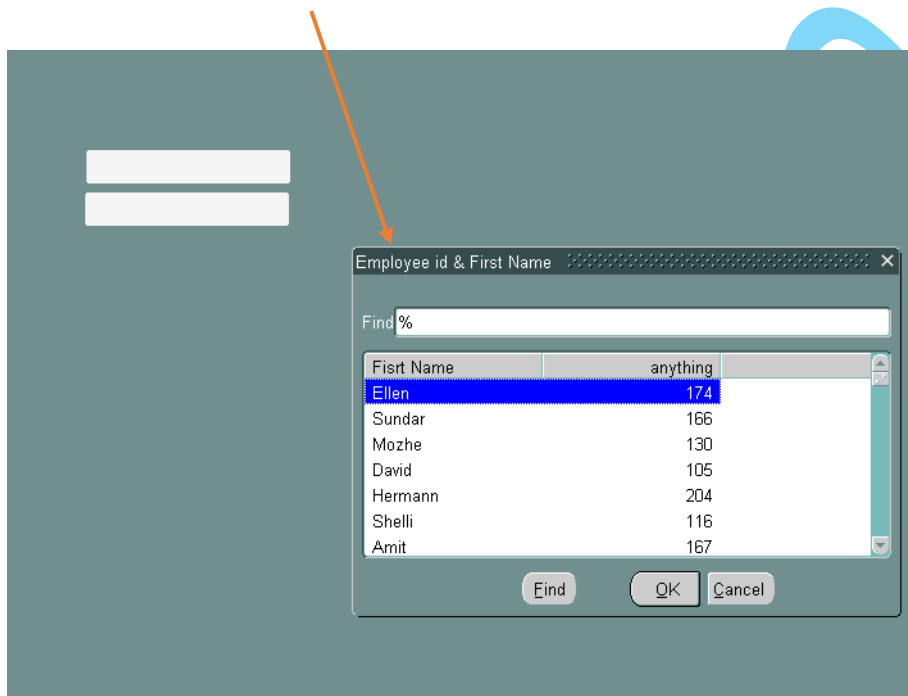


Update Only if NULL	No
Lock Record	No
List of Values (LOV)	
List of Values	EMPLOYEES
List X Position	<Null>
List Y Position	EMPLOYEES
Validate from List	No
Editor	
Editor	<Null>

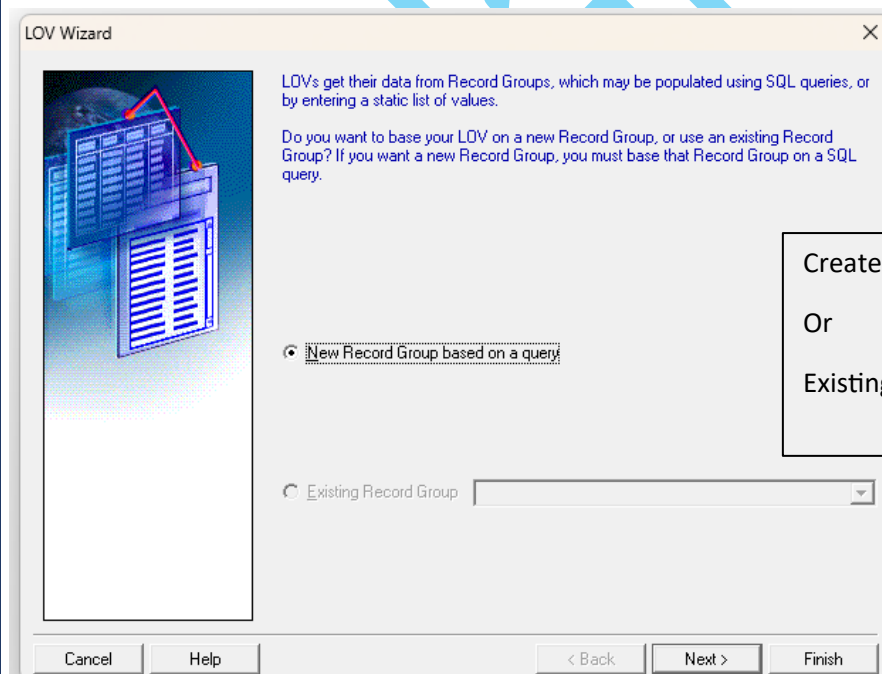
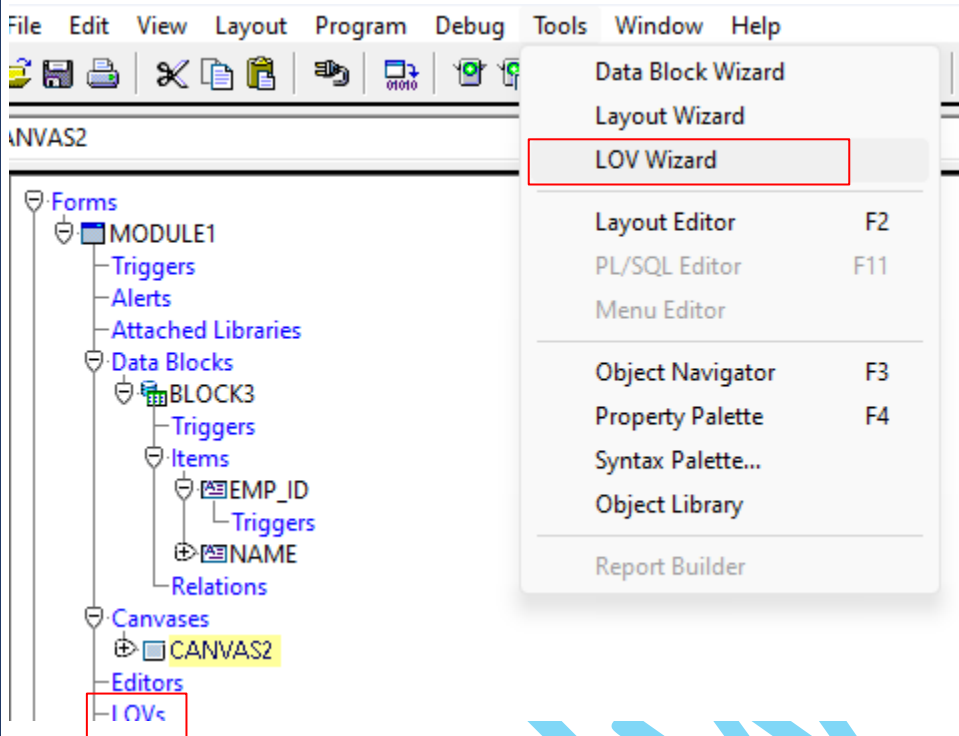
Run:



Click **CTRL-L**



- Use the LOV **wizard**.




Create new Record Group
Or
Existing Record group selection

There are three ways to write a ((select))

1

LOV Wizard



Record Groups can be based on SQL queries. Do you want to enter or modify the query that your LOV's Record Group uses?

If so, you may use the Oracle Developer Query Builder by clicking Build SQL Query. Or, you may enter your query directly into the SQL Query Statement field below.

Build SQL Query... Import SQL Query...

SQL Query Statement

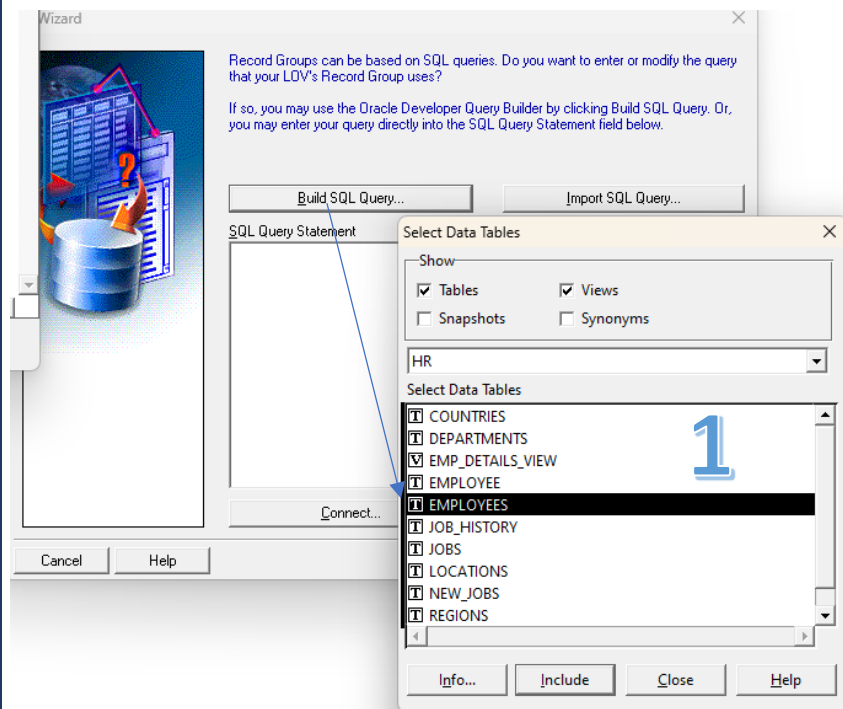
```
select employee_id, first_name from employees;
```

کتابتها مباشره

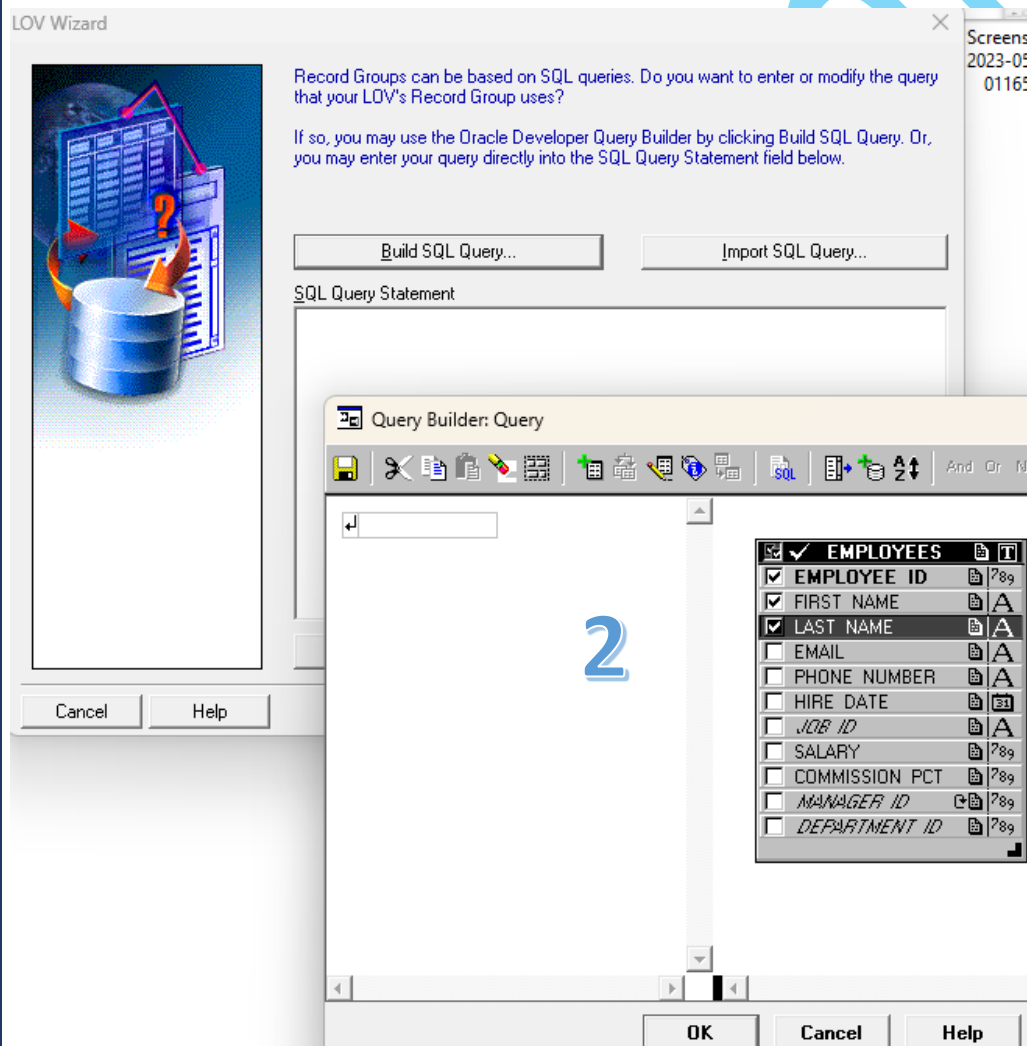
Connect... Check Syntax...

Cancel Help < Back Next > Finish

2

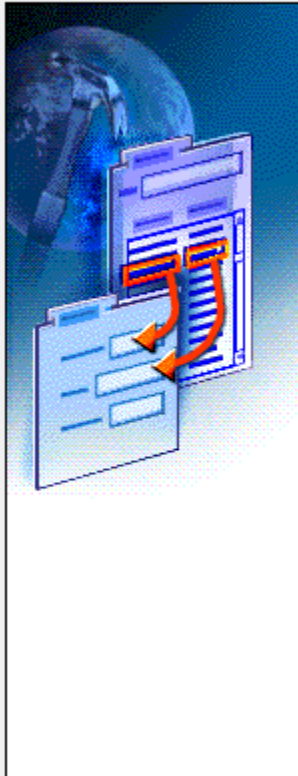


اختيار الجدول



اختيار الاعمدة

LOV Wizard



You are returning values from your LOV to the following items. You may assign your LOV to some or all of these items.

To which of these items do you wish to assign your LOV?

Return Items

Empty list box for Return Items.

Assigned Items

EMP_ID

اختر مكان ظهور القائمة



Cancel

Help

< Back

Next >

Finish

Window:

Emp id

name

Job

Name the job