The British College

KATHMANDU

Coursework Submission Coversheet
(individual coursework only)

Faculty of Arts, Environment and Technology

LBU Student Id: 77187980

FOR CHECKING BY THE STUDENT:

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: Bsc Hons Computing

Module code: CRN 16101

Module name: Advance Database B

Module run: 2017/18

Coursework title: **Database Server control and Performance**

Due Date: 2017/Oct/23

Module leader: (In LBU) Jackie Campbell, Sanela Lazarevski

Module tutor: (In TBC) Dibya Tara Shakya

TURNITIN Checked: YES [✓ NO] ***(please circle)***

Submission date& time: Date: 2017/Oct/23

Total Word Count: 2887 **Total Number of Pages (including this front sheet):40**

FOR COMPLETION BY THE FACULTY:

This mark is provisional and subject to moderation and approval by the relevant examining board

Teacher's Feedback

Teacher's Signature: ___________________ **Date: 10/23/2017**

TABLE OF CONTENTS

ASSIGNMENT 2: SERVER CONTROL AND PERFORMANCE

INTRODUCTION

Here In this assignment we are assigned to make a Server control database application, as required by task here we need to make a shareable and reusable code to build our application.

Here, I have made application named **Contractor Reporting System** using Oracle Apex 11g. In order to make this application I have used some tables from Placeu Database which are:

- Lds_Contractor
- Lds_Placement
- Lds_job_role
- Lds_previous_role

Contractor Reporting System application has used following oracle techniques:

1. Packages
2. Stored Procedures
3. Functions
4. Views

PACKAGES

Packages offer important features for reliable, maintainable, reusable code, often in team development efforts for large systems. Oracle supplies many PL/SQL packages with the Oracle server to extend database functionality and provide PL/SQL access to SQL features. Packages let you encapsulate logically related types, items, and subprograms in a named PL/SQL module. Each package is easy to understand, and the interfaces between packages are simple, clear, and well defined. This aids application development.

PACKAGE USED IN APPLICATION

- Pkg_final

STORED PROCEDURES

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs. Stored procedures can access or modify data in a database, but it is not tied to a specific database or object, which offers a number of advantages. A stored procedure provides an important layer of security between the user interface and the database.

PROCEDURES USED IN APPLICAION

- Prc_IUD
- Pro_IUD_Placement
- p_conPrev_role

FUNCTIONS

A SQL statement may use a standalone function or package function as an operator on one or more columns provided the function returns a valid Oracle database type.

What a user-defined function CAN and CANNOT do:

A function used within a SQL statement:

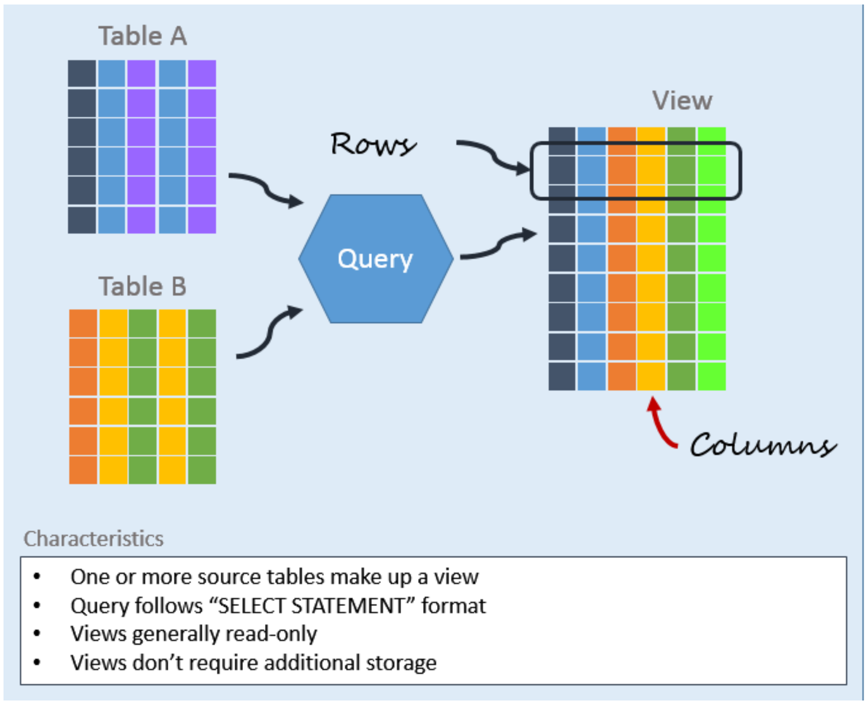
- may call other LOCAL or REMOTE stored procedures, i.e., other functions, procedures, or packages.
- may select from LOCAL or REMOTE database tables.
- may NOT directly or indirectly via other procedure call modify data in any table with an INSERT, UPDATE, or DELETE statement.

FUNCTION USED IN APPLICAION

- Check_position
- Find_con_skill
- find_contractor

VIEWS

A database view is a searchable object in a database that is defined by a query. Though a view doesn't store data, some refer to views as "virtual tables," you can query a view like you can a table. A view can combine data from two or more table, using joins, and also just contain a subset of information. This makes them convenient to abstract, or hide, complicated queries.



Characteristics

- One or more source tables make up a view
- Query follows "SELECT STATEMENT" format
- Views generally read-only
- Views don't require additional storage

VIEWS USED IN APPLICATION

- VIEW_CONTRACTOR_JOB_DESC

IMPLEMENTATION OF APPLICATION

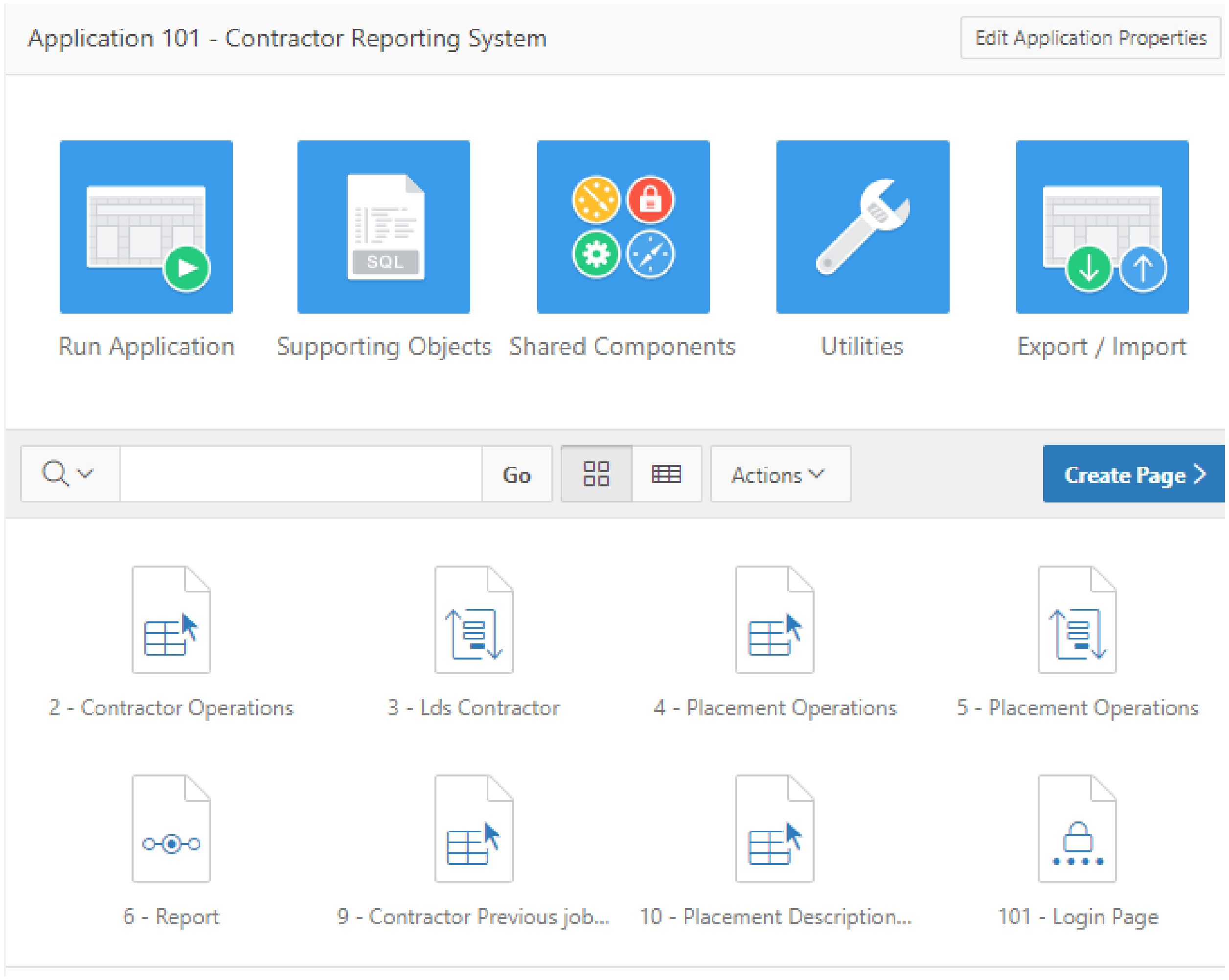


Fig: Start Page of the application

In order to enter towards application, we need to click run application which will take us to login page.

Here in My application user **kc** is a admin user, who will perform all the operations required for the application.

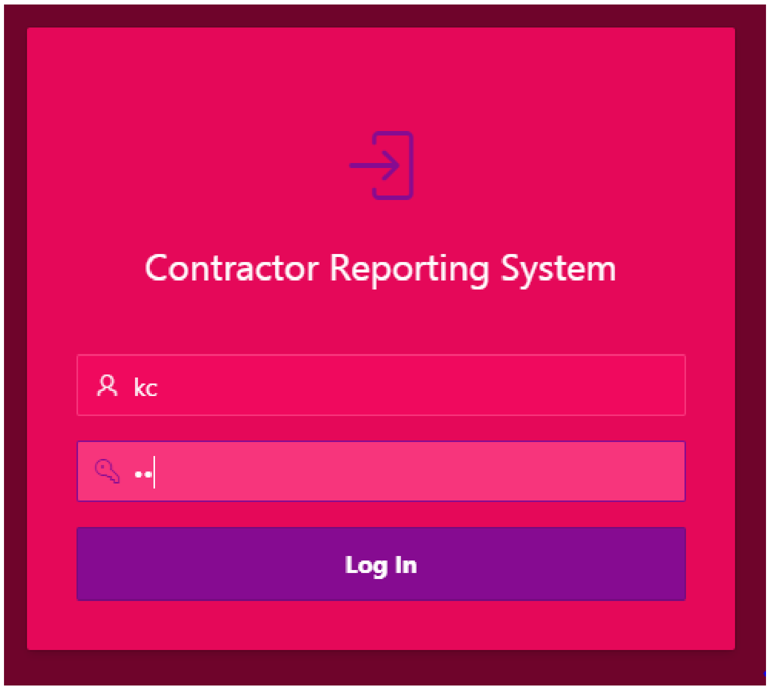
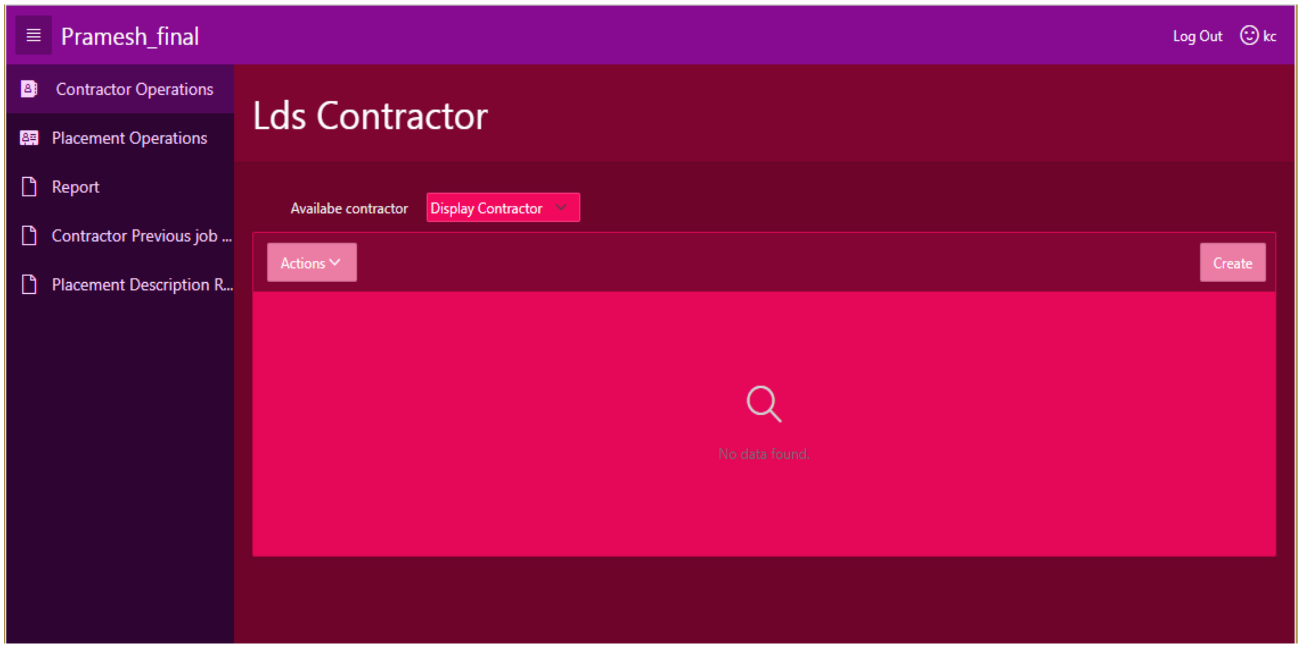


Fig: Login page of application

After login with suitable username and password application will enter towards operations/main page



IMPLEMENTATION OF PACKAGES, FUNCTIONS, STORED PROCEDURES AND VIEWS

Package Specification

```
create or replace package pkg_final as

 procedure Prc_IUD (operation_type varchar2,tble_name varchar2,name varchar2,pstcode varchar2,skill_one number,skill_two number,skill_three number,qualification number,assigned_role varchar2,id number);

procedure Pro_IUD_Placement(

v_PLACEMENT_ID number,

v_status varchar2,

v_table_name varchar2,

v_PLT_SHORT_DESC varchar2,

v_PLT_REQUIRED_START_DATE Date,

v_PLT_ESTIMATED_END_DATE DATE,

v_PTL_ACTUAL_START_DATE DATE,

v_PLT_ACTUAL_END_DATE DATE,

v_PLT_RENEWAL_NO NUMBER,

v_PLT_TO_PERMANENT VARCHAR2,

v_MAX_SALARY NUMBER,

v_MIN_SALARY NUMBER,

v_ACTUAL_SALARY NUMBER,

v_FK1_ACCOUNT_ID NUMBER,

v_FK2_CONTRACTOR_ID NUMBER,

v_FK3_JOB_ROLE_ID NUMBER);

procedure p_conPrev_role(v_con_id in number);

function check_position(v_plct_id in number) return varchar2;

function find_contractor(v_plct_id in number) return varchar2;

function find_con_skill(v_con_id in number) return varchar2;

function find_job_role(v_plc_id in number) return varchar2;

function find_placement_in_day(v_plct_id in number) return varchar2;

function find_conPrev_role(v_con_id in number) return varchar2;

 end;
```

Package Body

```
create or replace package body  pkg_final as

 procedure Prc_IUD (operation_type varchar2,tble_name varchar2,name varchar2,pstcode varchar2,skill_one
number,skill_two number,skill_three number,qualification number,assigned_role varchar2,id number)

is

query varchar2(3000);

begin

if upper(operation_type)=upper('insert') then

query:='insert into '||tble_name||' values ('||id||','||"" ||  " ||name||"" ||  "||','||"" ||  " ||pstcode||"" ||
"||','||skill_one||','||skill_two||','||skill_three||','||qualification||','||"" ||  " ||assigned_role||"" ||  "||')';

elsif upper(operation_type)=upper('update') then

query:='update '||tble_name||' set con_name='||"" ||  " ||name||"" ||  "||','||'CON_POSTCODE='||"" ||  " ||pstcode||"" ||
"||','||'CON_SKILL_1='||skill_one||','||'CON_SKILL_2='||skill_two||','||'CON_SKILL_3='||skill_three||','||'HIGHEST_Q
UAL='||qualification||','||'PREFERRED_ROLE='||"" ||  " ||assigned_role||"" ||  "||' where contractor_id='||id;

elsif upper(operation_type)=upper('delete') then

query:='delete from '||tble_name ||' Where contractor_id='||id;

end if;

execute immediate query;

end;


procedure Pro_IUD_Placement(

v_PLACEMENT_ID number,

v_status varchar2,

v_table_name varchar2,

v_PLT_SHORT_DESC varchar2,

v_PLT_REQUIRED_START_DATE Date,

v_PLT_ESTIMATED_END_DATE DATE,

v_PTL_ACTUAL_START_DATE DATE,

v_PLT_ACTUAL_END_DATE DATE,
v_PLT_RENEWAL_NO NUMBER,

v_PLT_TO_PERMANENT VARCHAR2,

v_MAX_SALARY NUMBER,

v_MIN_SALARY NUMBER,

v_ACTUAL_SALARY NUMBER,

v_FK1_ACCOUNT_ID NUMBER,

v_FK2_CONTRACTOR_ID NUMBER,

v_FK3_JOB_ROLE_ID NUMBER)

IS

Begin

        if v_status = 'insert' then

         execute immediate ' insert into ' || v_table_name || ' values (:a,:b,:c,:d,:e,:f,:g,:h,:i,:j,:k,:l,:m,:n) '

using v_PLACEMENT_ID ,v_PLT_SHORT_DESC ,v_PLT_REQUIRED_START_DATE
,v_PLT_ESTIMATED_END_DATE ,v_PTL_ACTUAL_START_DATE
,v_PLT_ACTUAL_END_DATE,v_PLT_RENEWAL_NO
,v_PLT_TO_PERMANENT,v_MAX_SALARY,v_MIN_SALARY,v_ACTUAL_SALARY
,v_FK1_ACCOUNT_ID,v_FK2_CONTRACTOR_ID ,v_FK3_JOB_ROLE_ID ;

               dbms_output.put_line('inserted');

         end if;

                if  v_status = 'update' then

        execute immediate ' update ' || v_table_name || ' set PLT_SHORT_DESC=
:a,PLT_REQUIRED_START_DATE=:b,PLT_ESTIMATED_END_DATE=
:c,PTL_ACTUAL_START_DATE=:d,PLT_ACTUAL_END_DATE=:e,PLT_RENEWAL_NO=:f,PLT_TO_PERM
ANENT=:g,MAX_SALARY=:h,MIN_SALARY=:i,ACTUAL_SALARY=:j,FK1_ACCOUNT_ID=:k,FK2_CONT
RACTOR_ID=:l,FK3_JOB_ROLE_ID=:m where PLACEMENT_ID = :n'

 using v_PLT_SHORT_DESC ,v_PLT_REQUIRED_START_DATE ,v_PLT_ESTIMATED_END_DATE
,v_PTL_ACTUAL_START_DATE ,v_PLT_ACTUAL_END_DATE,v_PLT_RENEWAL_NO
,v_PLT_TO_PERMANENT,v_MAX_SALARY,v_MIN_SALARY,v_ACTUAL_SALARY
,v_FK1_ACCOUNT_ID,v_FK2_CONTRACTOR_ID,v_FK3_JOB_ROLE_ID,v_PLACEMENT_ID ;

              dbms_output.put_line('updated');
```

```
    elsif v_status = 'delete' then

     execute immediate ' delete from ' || v_table_name || ' where PLACEMENT_ID = :a '

     using v_PLACEMENT_ID ;

            dbms_output.put_line('deleted');

       end if;

end;


procedure p_conPrev_role(v_con_id in number)

is

v_s_year number;

v_s_month number;

v_e_year number;

v_e_month number;

v_pre_job varchar2(50);

v_con varchar2(50);

v_total_yr  number;

cursor cr is select
to_number(to_char(start_date,'YYYY')),to_number(to_char(start_date,'mm')),to_number(to_char(end_date,'YYYY')
),to_number(to_char(end_date,'mm')),job_role_name,con_name  from lds_previous_roles inner join lds_contractor
on lds_previous_roles.fk1_contractor_id=lds_contractor.contractor_id where

lds_contractor.contractor_id=v_con_id;

begin

open cr;

loop

fetch cr into v_s_year ,v_s_month ,v_e_year ,v_e_month ,v_pre_job ,v_con ;

exit when cr%notfound;

if v_s_year <> v_e_year then

 v_total_yr :=v_e_year-v_s_year;

elsif v_s_year = v_e_year then
```

```
raise_application_error(-20001,'Contractor has worked less than a year');

end if;

end loop;

close cr;

end;


function check_position(v_plct_id in number) return varchar2

is

v_position varchar2(55);

v_plt char(5);

v_salary number;

begin

select plt_to_permanent,actual_salary into v_plt,v_salary  from lds_placement where placement_id = v_plct_id;

if v_plt='N' then

v_position:='placement position  for ID= '||v_plct_id|| ' is Temporary';

elsif v_plt='Y' then

v_position:='placement position  for ID= '||v_plct_id|| ' is Permanent';

end if;

dbms_output.put_line(v_position );

return v_position;

end;



function find_contractor(v_plct_id in number) return varchar2

is

v_con_name  lds_contractor.con_name%type;

v_CONTRACTOR_ID number;
```

```
begin

select con_name Contractor, CONTRACTOR_ID ID into v_con_name,v_CONTRACTOR_ID  from lds_contractor
where contractor_id=(select FK2_CONTRACTOR_ID from lds_placement where placement_id =v_plct_id);

return  'Contractor ID= '||v_CONTRACTOR_ID ||', Contractor = '||v_con_name;

end;

function find_con_skill(v_con_id in number) return varchar2

is

v_skill1 varchar2(50);

v_skill2 varchar2(50);

v_skill3 varchar2(50);

v_con_name varchar2(50);

begin

select s.skill_desc,c.con_name into v_skill1,v_con_name  from lds_skill s inner join lds_contractor c on
s.skill_id=c.con_skill_1 where c.contractor_id=v_con_id;

select s.skill_desc,c.con_name into v_skill2,v_con_name  from lds_skill s inner join lds_contractor c on
s.skill_id=c.con_skill_2 where c.contractor_id=v_con_id;

select s.skill_desc,c.con_name into v_skill3,v_con_name  from lds_skill s inner join lds_contractor c on
s.skill_id=c.con_skill_3 where c.contractor_id=v_con_id;

return 'Contractor Name = '||v_con_name||' ,  Skill one = '||v_skill1||',    Skill Two = '||v_skill2||' ,   Skill Three =
'||v_skill3;

end;


function find_conPrev_role(v_con_id in number) return varchar2

is

v_s_year number;

v_s_month number;

v_e_year number;

v_e_month number;

v_pre_job varchar2(50);

v_con varchar2(50);
```

```
cursor cr is select
to_number(to_char(start_date,'YYYY')),to_number(to_char(start_date,'mm')),to_number(to_char(end_date,'YYYY')
),to_number(to_char(end_date,'mm')),job_role_name,con_name  from lds_previous_roles inner join lds_contractor
on lds_previous_roles.fk1_contractor_id=lds_contractor.contractor_id where

lds_contractor.contractor_id=v_con_id;

begin

open cr;

loop

fetch cr into v_s_year ,v_s_month ,v_e_year ,v_e_month ,v_pre_job ,v_con ;

exit when cr%notfound;

return 'Previous Job role of '||v_con||' is '||v_pre_job||', from '||v_s_year||' to '||v_e_year;

end loop;

close cr;

end;

 end  pkg_final;
```

In above Package I have stored all the required procedures and functions to implement in applications.

CRUD OPERATIONS FOR LDS_CONTRACTOR

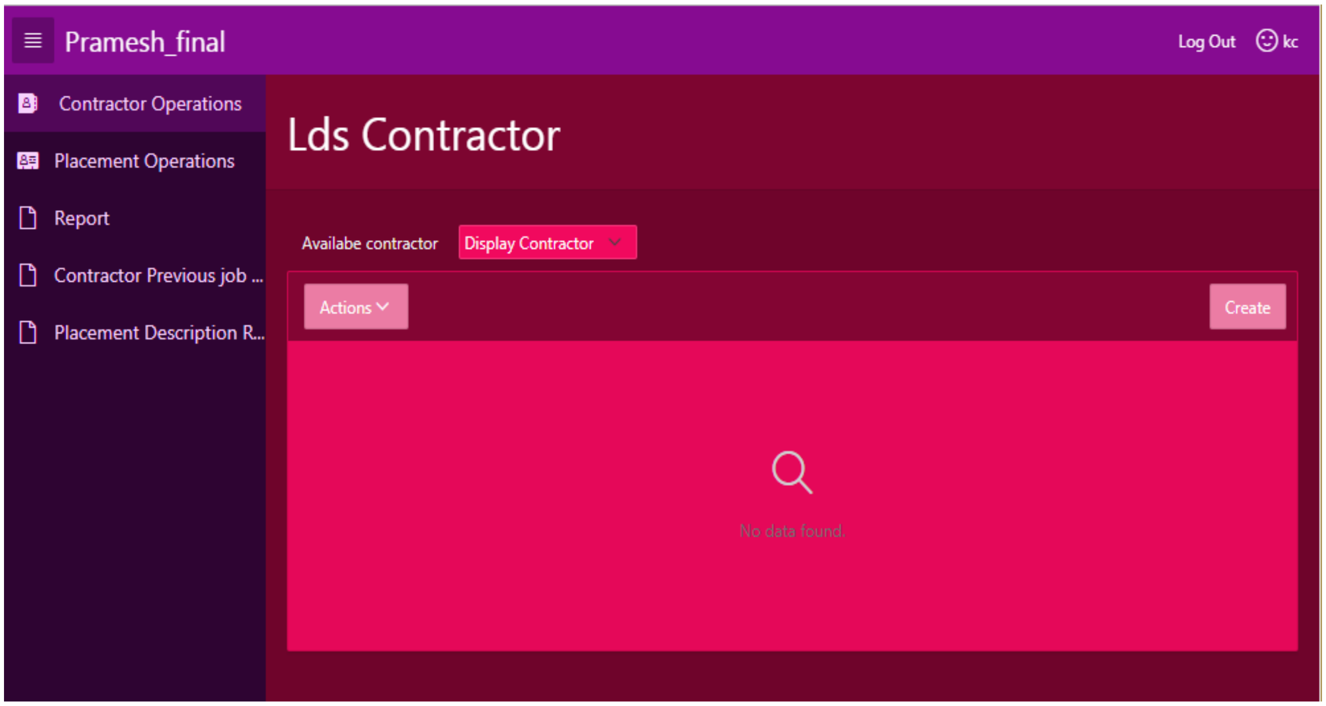
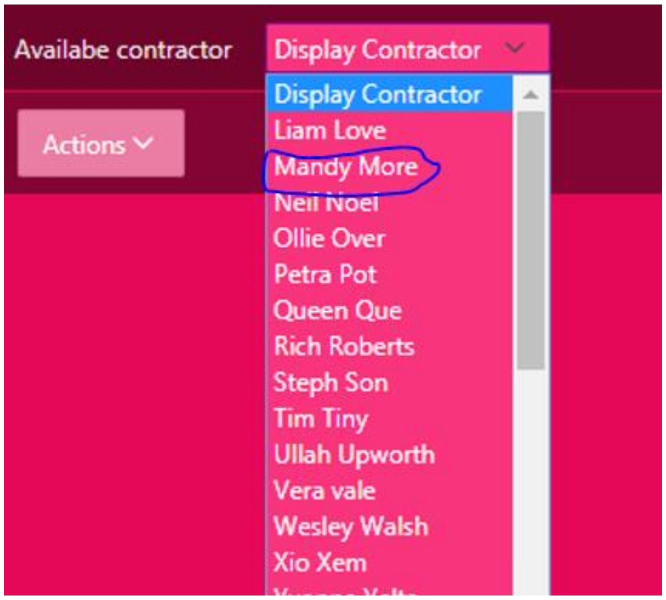
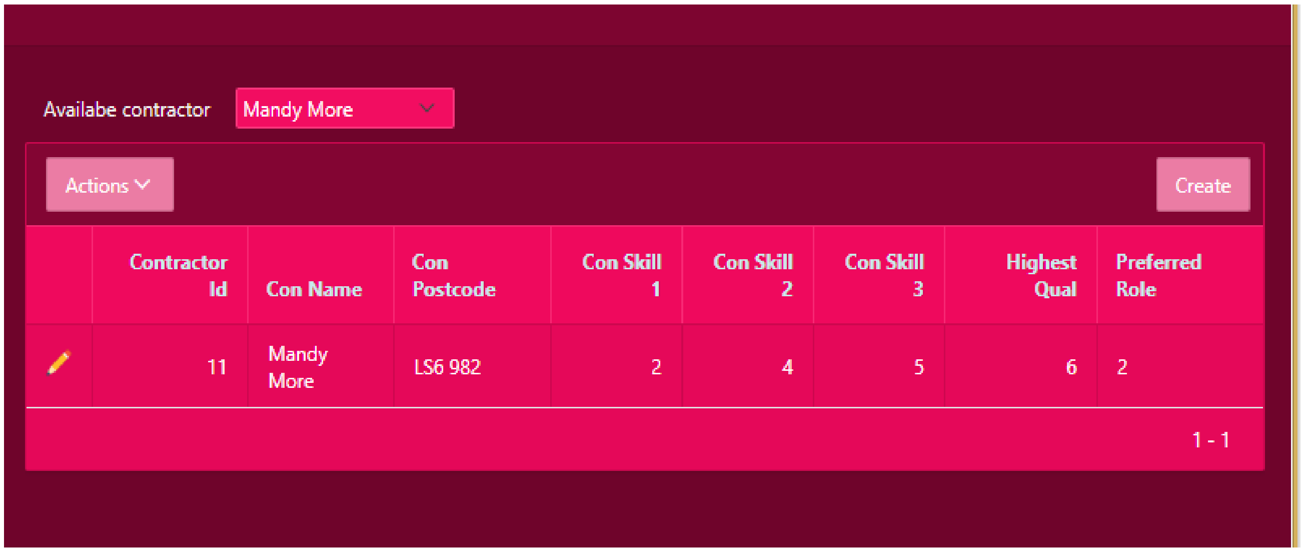


Fig: Contractor Operations Page

In Order to view the contractor details we need to select the values from the combo box as follows.



Above Contractor Mandy More is selected from the combo box so Mandy More details will show in table.

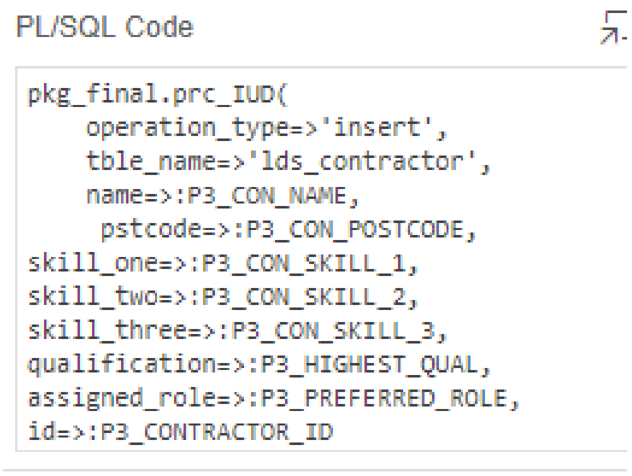


	Contractor Id	Con Name	Con Postcode	Con Skill 1	Con Skill 2	Con Skill 3	Highest Qual	Preferred Role
	11	Mandy More	LS6 982	2	4	5	6	2

1 - 1

Fig: Mandy More details are shown in table

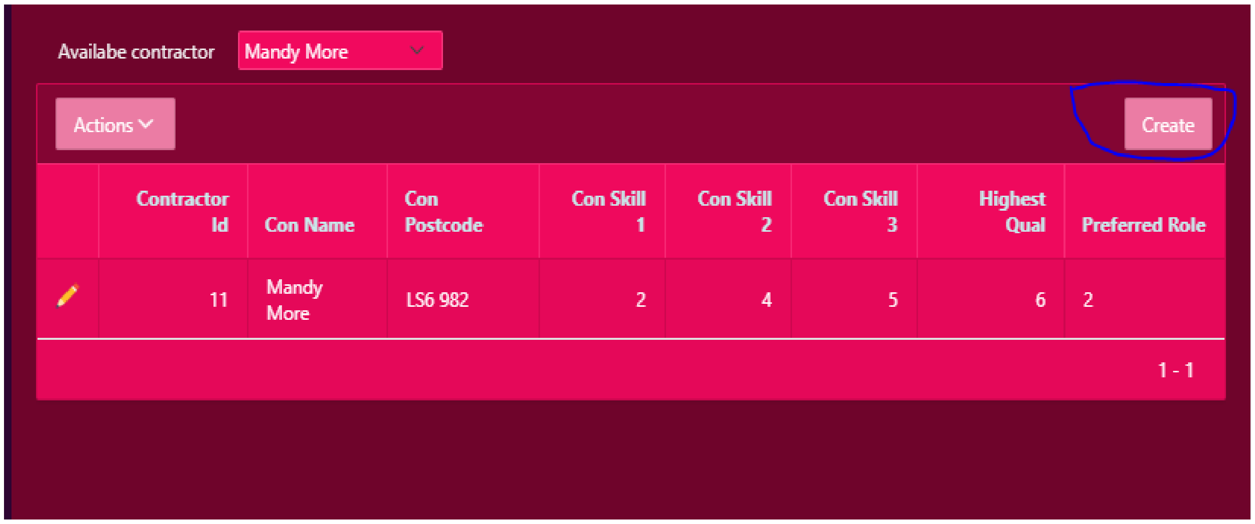
In order to add, update and delete contractor, I have made process where I have call the package procedure which will insert update and delete contractor.



```
pkg_final.prc_IUD(
    operation_type=>'insert',
    tble_name=>'lds_contractor',
    name=>:P3_CON_NAME,
     pstcode=>:P3_CON_POSTCODE,
skill_one=>:P3_CON_SKILL_1,
skill_two=>:P3_CON_SKILL_2,
skill_three=>:P3_CON_SKILL_3,
qualification=>:P3_HIGHEST_QUAL,
assigned_role=>:P3_PREFERRED_ROLE,
id=>:P3_CONTRACTOR_ID
```

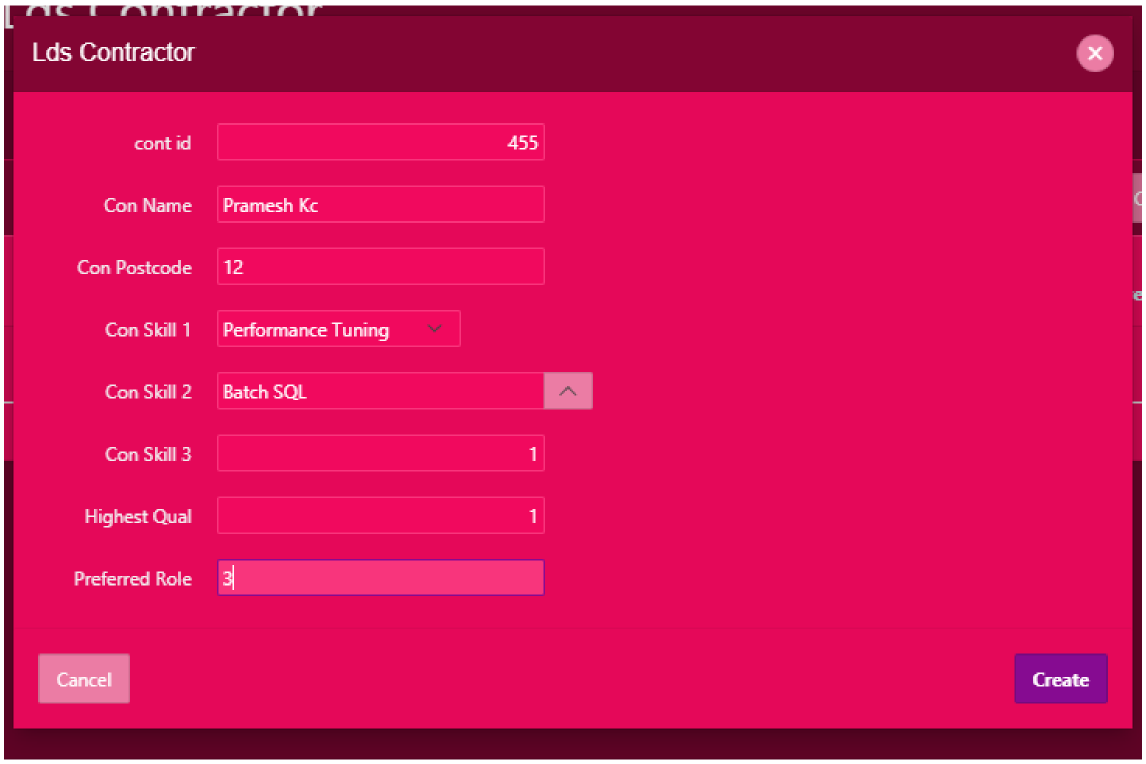
Implementation of procedure and package to insert contractor. Evidence are shown below.

In order to add Contractor firstly we need to click create button and pop up box will appear where we need to fill up the form to add contractor.



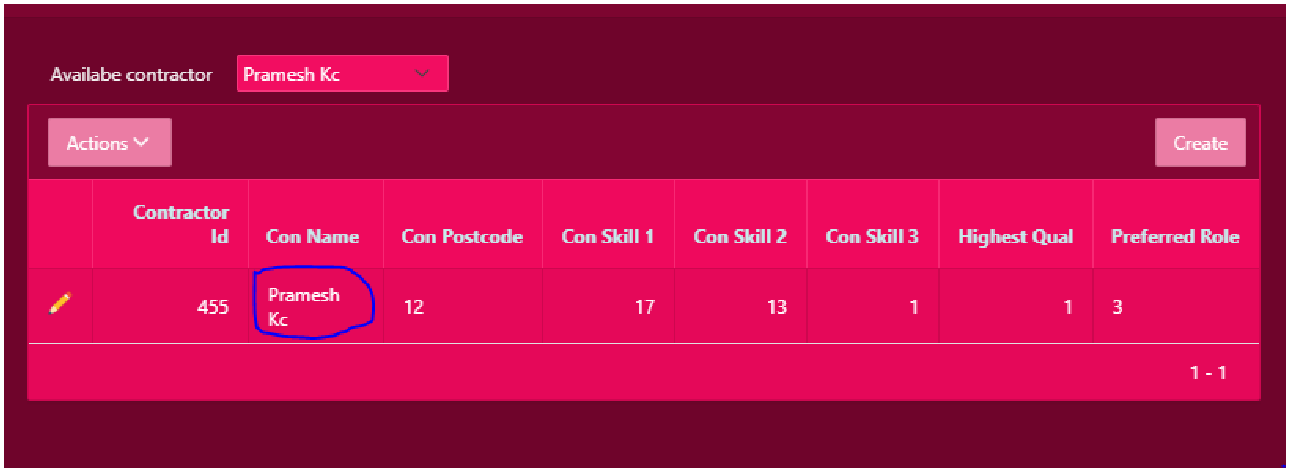
	Contractor Id	Con Name	Con Postcode	Con Skill 1	Con Skill 2	Con Skill 3	Highest Qual	Preferred Role
	11	Mandy More	LS6 982	2	4	5	6	2

1 - 1



cont id	455
Con Name	Pramesh Kc
Con Postcode	12
Con Skill 1	Performance Tuning
Con Skill 2	Batch SQL
Con Skill 3	1
Highest Qual	1
Preferred Role	3

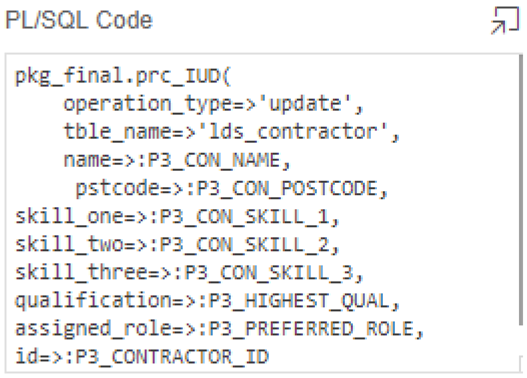
After filling the form need to click create button.



	Contractor Id	Con Name	Con Postcode	Con Skill 1	Con Skill 2	Con Skill 3	Highest Qual	Preferred Role
✎	455	Pramesh Kc	12	17	13	1	1	3

Fig: contractor Added successfully

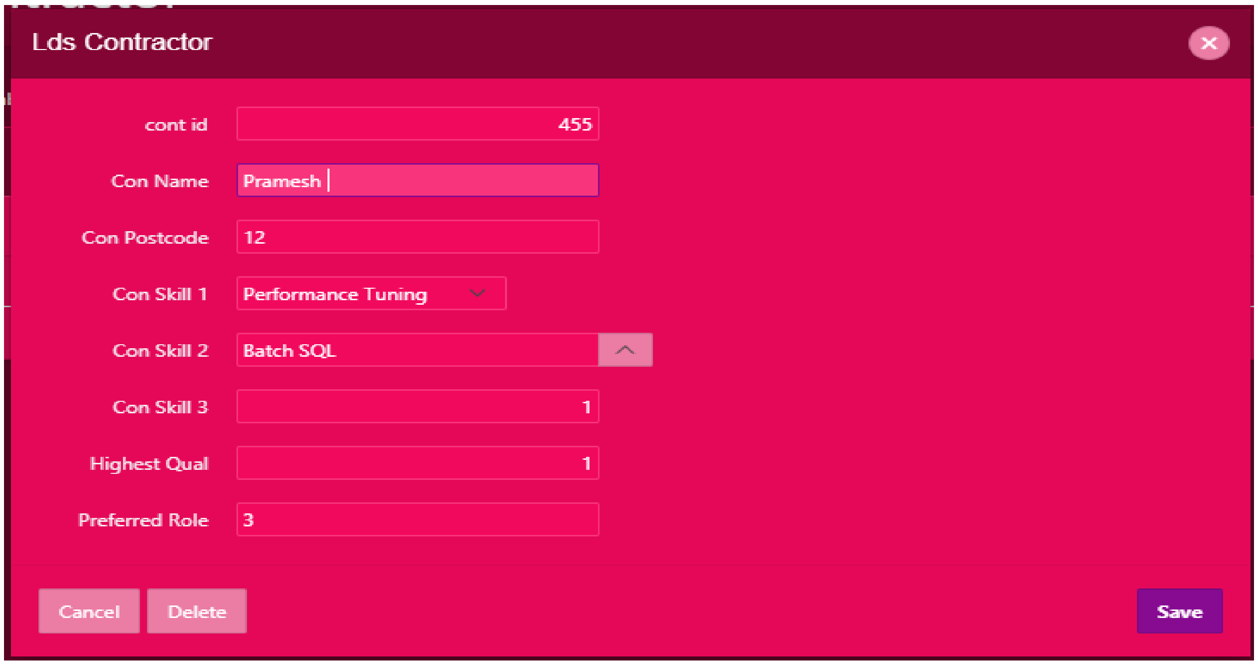
In order to update contractor, we need to click at the pencil logo which is in left side of the table.



```
pkg_final.prc_IUD(
    operation_type=>'update',
    tble_name=>'lds_contractor',
    name=>:P3_CON_NAME,
     pstcode=>:P3_CON_POSTCODE,
skill_one=>:P3_CON_SKILL_1,
skill_two=>:P3_CON_SKILL_2,
skill_three=>:P3_CON_SKILL_3,
qualification=>:P3_HIGHEST_QUAL,
assigned_role=>:P3_PREFERRED_ROLE,
id=>:P3_CONTRACTOR_ID
```

Implementation of procedure and package to update contractor. Evidence are shown below.

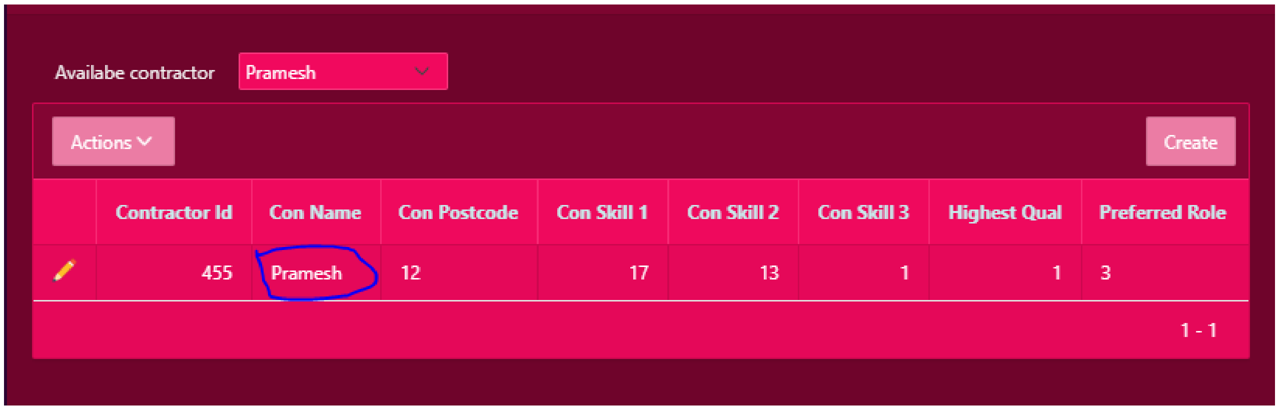
Here I’m going to update Pramesh Kc to only pramesh



Lds Contractor

cont id	455
Con Name	Pramesh
Con Postcode	12
Con Skill 1	Performance Tuning
Con Skill 2	Batch SQL
Con Skill 3	1
Highest Qual	1
Preferred Role	3

We need to click save button in order to update.

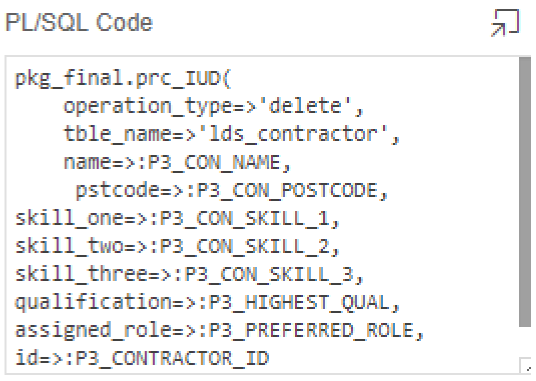


Availabe contractor: Pramesh

	Contractor Id	Con Name	Con Postcode	Con Skill 1	Con Skill 2	Con Skill 3	Highest Qual	Preferred Role
	455	Pramesh	12	17	13	1	1	3

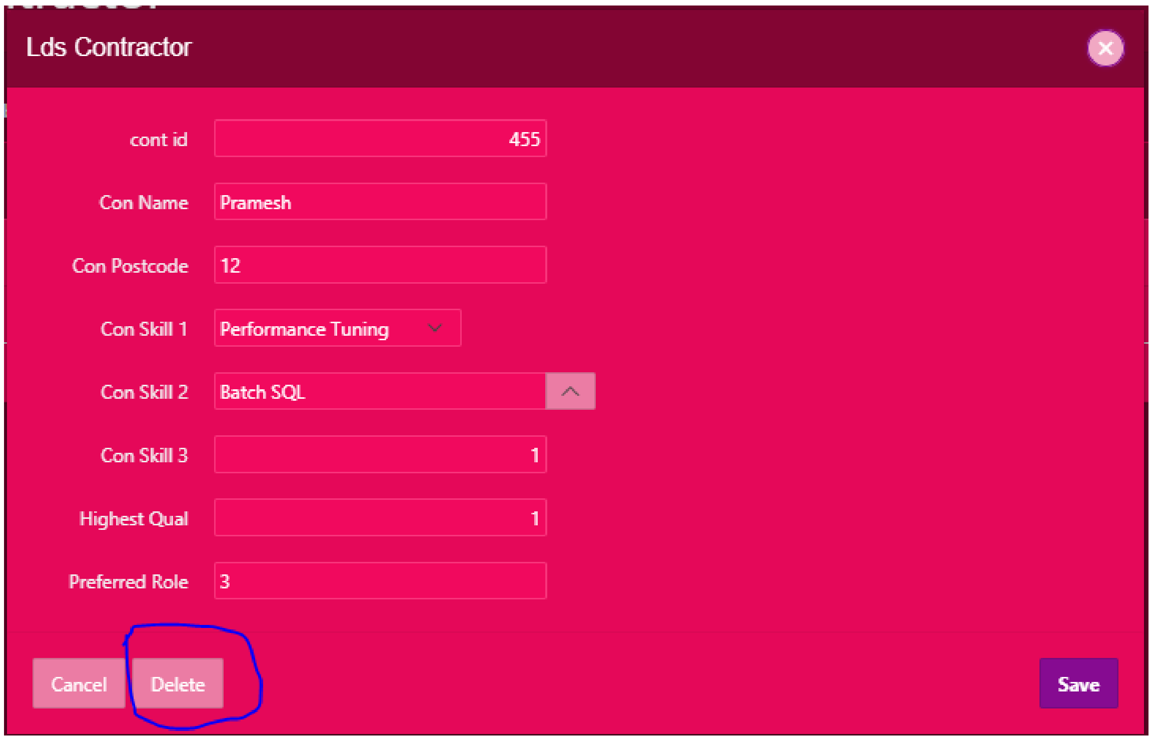
1 - 1

Fig: contractor has been updated successfully



```
pkg_final.prc_IUD(
    operation_type=>'delete',
    tble_name=>'lds_contractor',
    name=>:P3_CON_NAME,
     pstcode=>:P3_CON_POSTCODE,
skill_one=>:P3_CON_SKILL_1,
skill_two=>:P3_CON_SKILL_2,
skill_three=>:P3_CON_SKILL_3,
qualification=>:P3_HIGHEST_QUAL,
assigned_role=>:P3_PREFERRED_ROLE,
id=>:P3_CONTRACTOR_ID
```

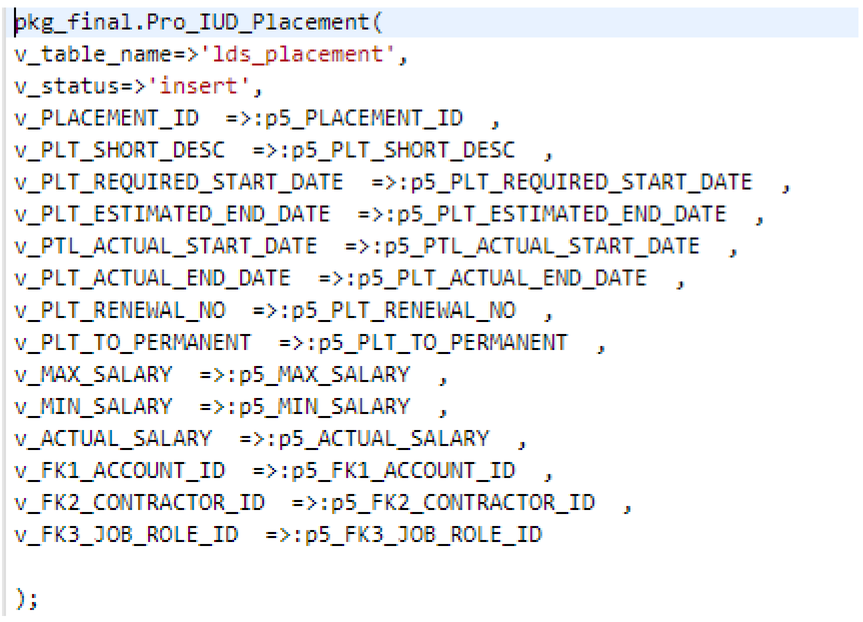
Fig: procedure and package to delete contractor



cont id	455
Con Name	Pramesh
Con Postcode	12
Con Skill 1	Performance Tuning
Con Skill 2	Batch SQL
Con Skill 3	1
Highest Qual	1
Preferred Role	3

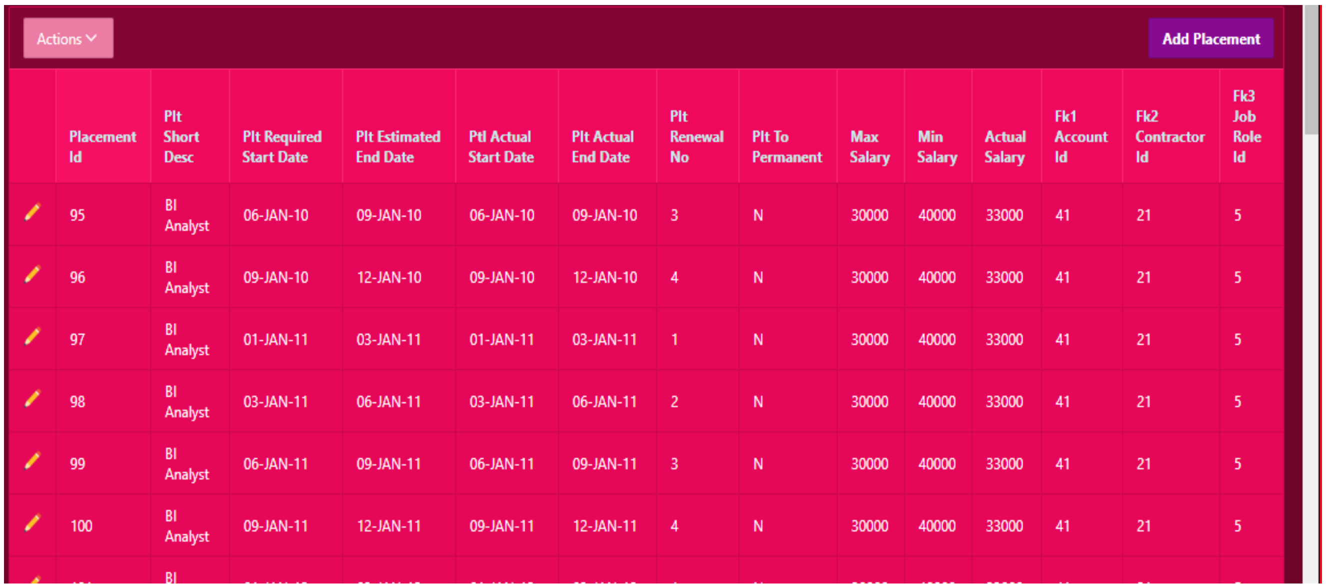
In order to delete contractor, we need click delete button.

CRUD OPERATION FOR LDS_PLACEMENT



```
pkg_final.Pro_IUD_Placement(
v_table_name=>'lds_placement',
v_status=>'insert',
v_PLACEMENT_ID  =>:p5_PLACEMENT_ID  ,
v_PLT_SHORT_DESC  =>:p5_PLT_SHORT_DESC  ,
v_PLT_REQUIRED_START_DATE  =>:p5_PLT_REQUIRED_START_DATE  ,
v_PLT_ESTIMATED_END_DATE  =>:p5_PLT_ESTIMATED_END_DATE  ,
v_PTL_ACTUAL_START_DATE  =>:p5_PTL_ACTUAL_START_DATE  ,
v_PLT_ACTUAL_END_DATE  =>:p5_PLT_ACTUAL_END_DATE  ,
v_PLT_RENEWAL_NO  =>:p5_PLT_RENEWAL_NO  ,
v_PLT_TO_PERMANENT  =>:p5_PLT_TO_PERMANENT  ,
v_MAX_SALARY  =>:p5_MAX_SALARY  ,
v_MIN_SALARY  =>:p5_MIN_SALARY  ,
v_ACTUAL_SALARY  =>:p5_ACTUAL_SALARY  ,
v_FK1_ACCOUNT_ID  =>:p5_FK1_ACCOUNT_ID  ,
v_FK2_CONTRACTOR_ID  =>:p5_FK2_CONTRACTOR_ID  ,
v_FK3_JOB_ROLE_ID  =>:p5_FK3_JOB_ROLE_ID

);
```

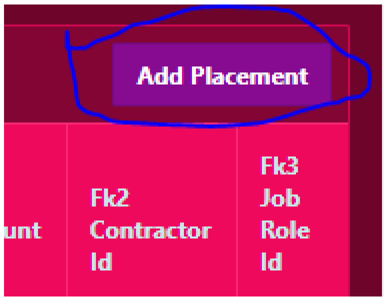
Fig: implementation of package and procedure to add placement

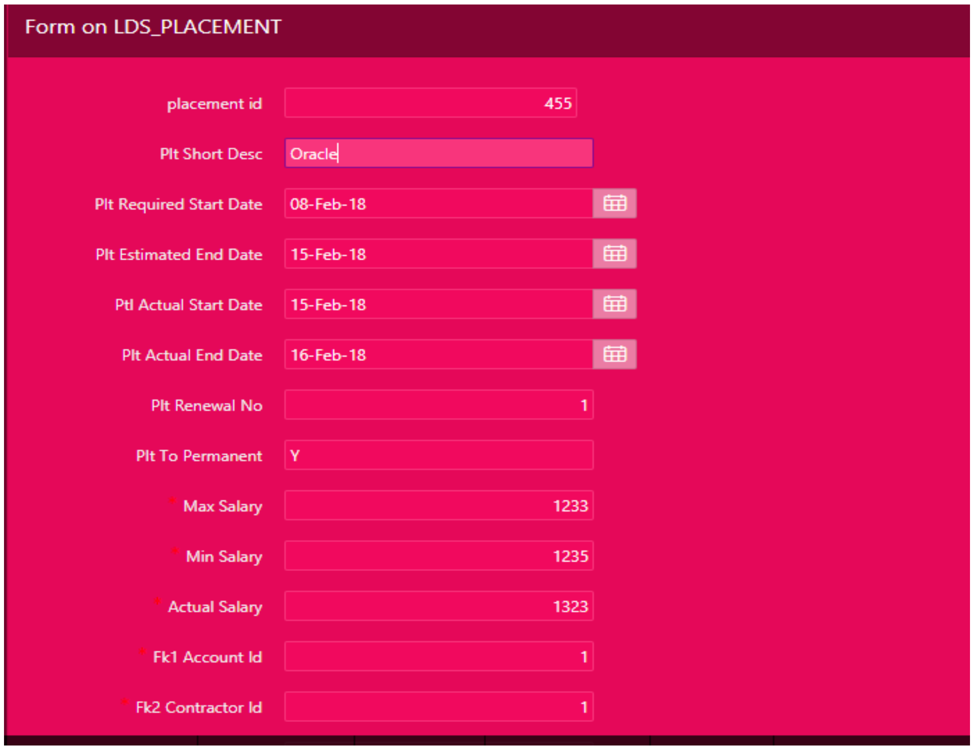


	Placement Id	Plt Short Desc	Plt Required Start Date	Plt Estimated End Date	Ptl Actual Start Date	Plt Actual End Date	Plt Renewal No	Plt To Permanent	Max Salary	Min Salary	Actual Salary	Fk1 Account Id	Fk2 Contractor Id	Fk3 Job Role Id
✎	95	BI Analyst	06-JAN-10	09-JAN-10	06-JAN-10	09-JAN-10	3	N	30000	40000	33000	41	21	5
✎	96	BI Analyst	09-JAN-10	12-JAN-10	09-JAN-10	12-JAN-10	4	N	30000	40000	33000	41	21	5
✎	97	BI Analyst	01-JAN-11	03-JAN-11	01-JAN-11	03-JAN-11	1	N	30000	40000	33000	41	21	5
✎	98	BI Analyst	03-JAN-11	06-JAN-11	03-JAN-11	06-JAN-11	2	N	30000	40000	33000	41	21	5
✎	99	BI Analyst	06-JAN-11	09-JAN-11	06-JAN-11	09-JAN-11	3	N	30000	40000	33000	41	21	5
✎	100	BI Analyst	09-JAN-11	12-JAN-11	09-JAN-11	12-JAN-11	4	N	30000	40000	33000	41	21	5
		BI												

Fig: Placement Operation Page

In order to add placement firstly we need to click create button and pop up box will appear where we need to fill up the form to add placement.

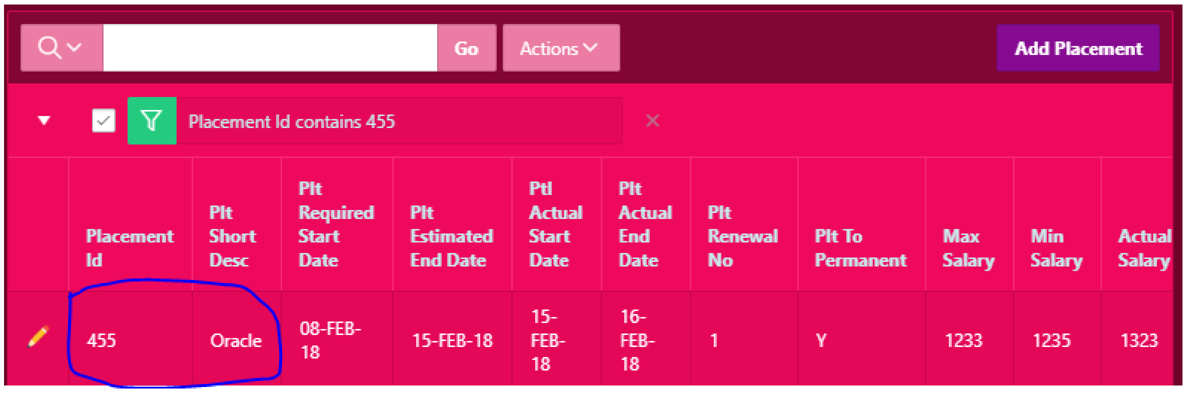




Form on LDS_PLACEMENT

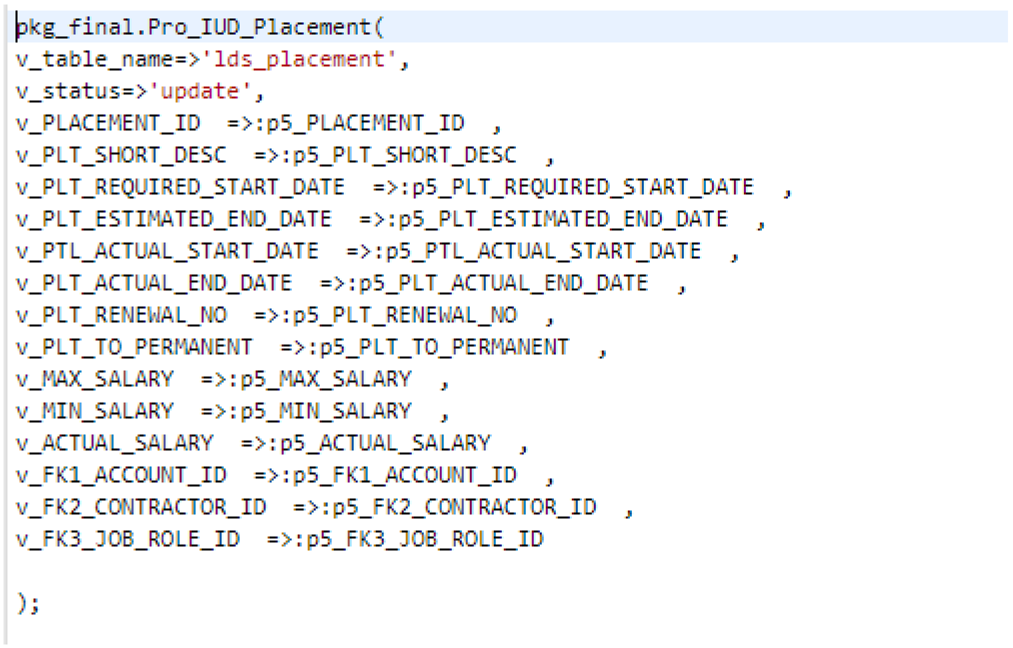
placement id	455
Plt Short Desc	Oracle
Plt Required Start Date	08-Feb-18
Plt Estimated End Date	15-Feb-18
Ptl Actual Start Date	15-Feb-18
Plt Actual End Date	16-Feb-18
Plt Renewal No	1
Plt To Permanent	Y
* Max Salary	1233
* Min Salary	1235
* Actual Salary	1323
* Fk1 Account Id	1
* Fk2 Contractor Id	1

After filling the form need to click create button.



Placement Id	Plt Short Desc	Plt Required Start Date	Plt Estimated End Date	Ptl Actual Start Date	Plt Actual End Date	Plt Renewal No	Plt To Permanent	Max Salary	Min Salary	Actual Salary
455	Oracle	08-FEB-18	15-FEB-18	15-FEB-18	16-FEB-18	1	Y	1233	1235	1323

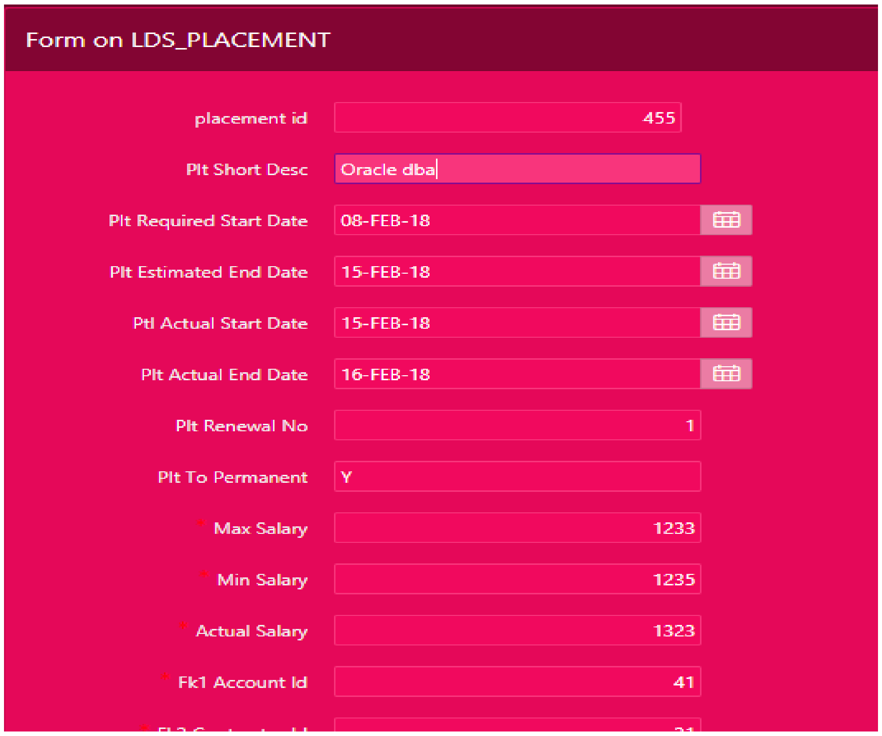
Fig: Placement added successfully



```
pkg_final.Pro_IUD_Placement(
v_table_name=>'lds_placement',
v_status=>'update',
v_PLACEMENT_ID  =>:p5_PLACEMENT_ID  ,
v_PLT_SHORT_DESC  =>:p5_PLT_SHORT_DESC  ,
v_PLT_REQUIRED_START_DATE  =>:p5_PLT_REQUIRED_START_DATE  ,
v_PLT_ESTIMATED_END_DATE  =>:p5_PLT_ESTIMATED_END_DATE  ,
v_PTL_ACTUAL_START_DATE  =>:p5_PTL_ACTUAL_START_DATE  ,
v_PLT_ACTUAL_END_DATE  =>:p5_PLT_ACTUAL_END_DATE  ,
v_PLT_RENEWAL_NO  =>:p5_PLT_RENEWAL_NO  ,
v_PLT_TO_PERMANENT  =>:p5_PLT_TO_PERMANENT  ,
v_MAX_SALARY  =>:p5_MAX_SALARY  ,
v_MIN_SALARY  =>:p5_MIN_SALARY  ,
v_ACTUAL_SALARY  =>:p5_ACTUAL_SALARY  ,
v_FK1_ACCOUNT_ID  =>:p5_FK1_ACCOUNT_ID  ,
v_FK2_CONTRACTOR_ID  =>:p5_FK2_CONTRACTOR_ID  ,
v_FK3_JOB_ROLE_ID  =>:p5_FK3_JOB_ROLE_ID

);
```

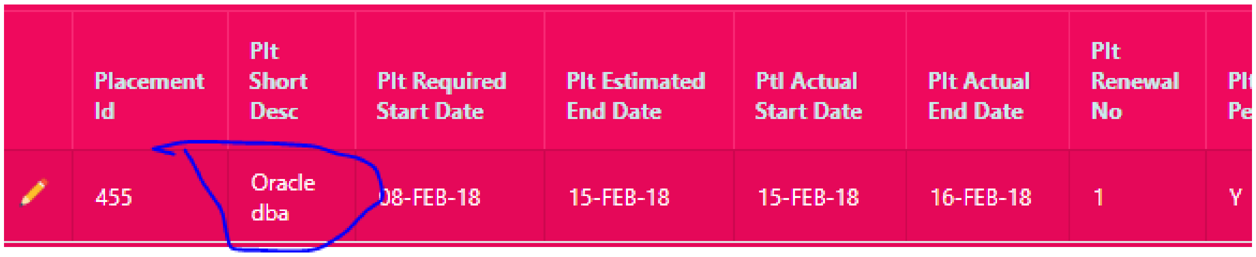
Fig: implementation of package and procedure to update placement

Here I'm going to update placement oracle as oracle dba



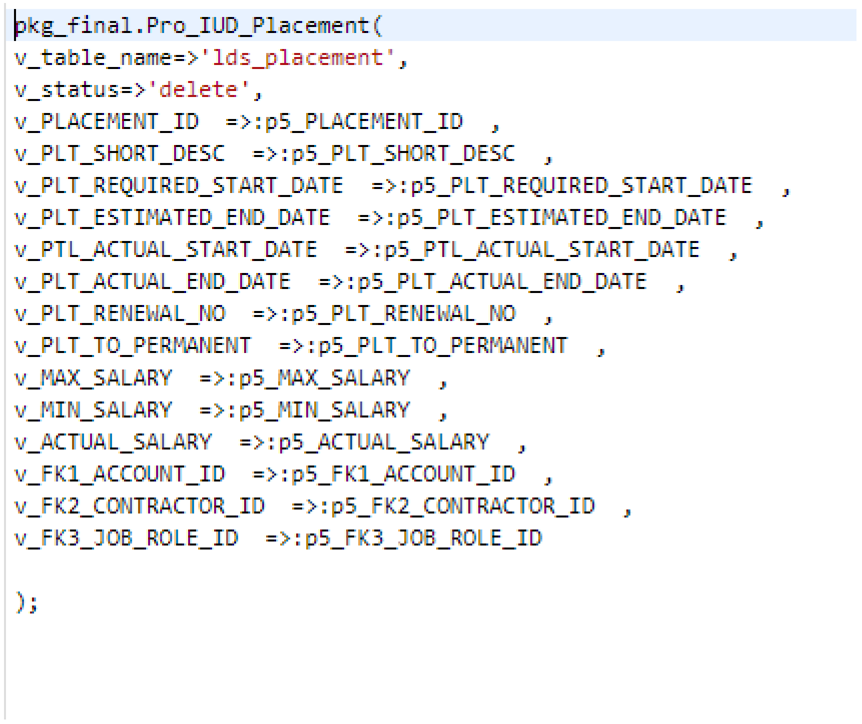
Form on LDS_PLACEMENT

placement id	455
Plt Short Desc	Oracle dba
Plt Required Start Date	08-FEB-18
Plt Estimated End Date	15-FEB-18
Ptl Actual Start Date	15-FEB-18
Plt Actual End Date	16-FEB-18
Plt Renewal No	1
Plt To Permanent	Y
* Max Salary	1233
* Min Salary	1235
* Actual Salary	1323
* Fk1 Account Id	41



	Placement Id	Plt Short Desc	Plt Required Start Date	Plt Estimated End Date	Ptl Actual Start Date	Plt Actual End Date	Plt Renewal No	Plt Pe
✎	455	Oracle dba	08-FEB-18	15-FEB-18	15-FEB-18	16-FEB-18	1	Y

Fig: placement updated successfully

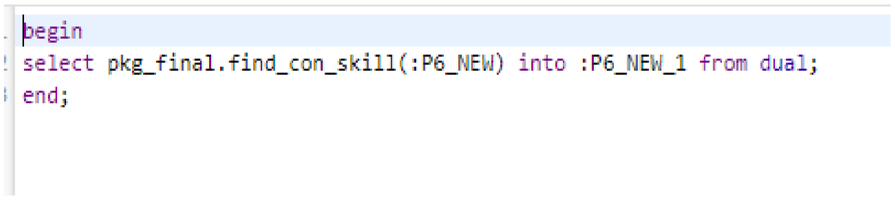


```
pkg_final.Pro_IUD_Placement(
v_table_name=>'lds_placement',
v_status=>'delete',
v_PLACEMENT_ID  =>:p5_PLACEMENT_ID  ,
v_PLT_SHORT_DESC  =>:p5_PLT_SHORT_DESC  ,
v_PLT_REQUIRED_START_DATE  =>:p5_PLT_REQUIRED_START_DATE  ,
v_PLT_ESTIMATED_END_DATE  =>:p5_PLT_ESTIMATED_END_DATE  ,
v_PTL_ACTUAL_START_DATE  =>:p5_PTL_ACTUAL_START_DATE  ,
v_PLT_ACTUAL_END_DATE  =>:p5_PLT_ACTUAL_END_DATE  ,
v_PLT_RENEWAL_NO  =>:p5_PLT_RENEWAL_NO  ,
v_PLT_TO_PERMANENT  =>:p5_PLT_TO_PERMANENT  ,
v_MAX_SALARY  =>:p5_MAX_SALARY  ,
v_MIN_SALARY  =>:p5_MIN_SALARY  ,
v_ACTUAL_SALARY  =>:p5_ACTUAL_SALARY  ,
v_FK1_ACCOUNT_ID  =>:p5_FK1_ACCOUNT_ID  ,
v_FK2_CONTRACTOR_ID  =>:p5_FK2_CONTRACTOR_ID  ,
v_FK3_JOB_ROLE_ID  =>:p5_FK3_JOB_ROLE_ID

);
```

Fig: implementation of package and procedure to delete placement

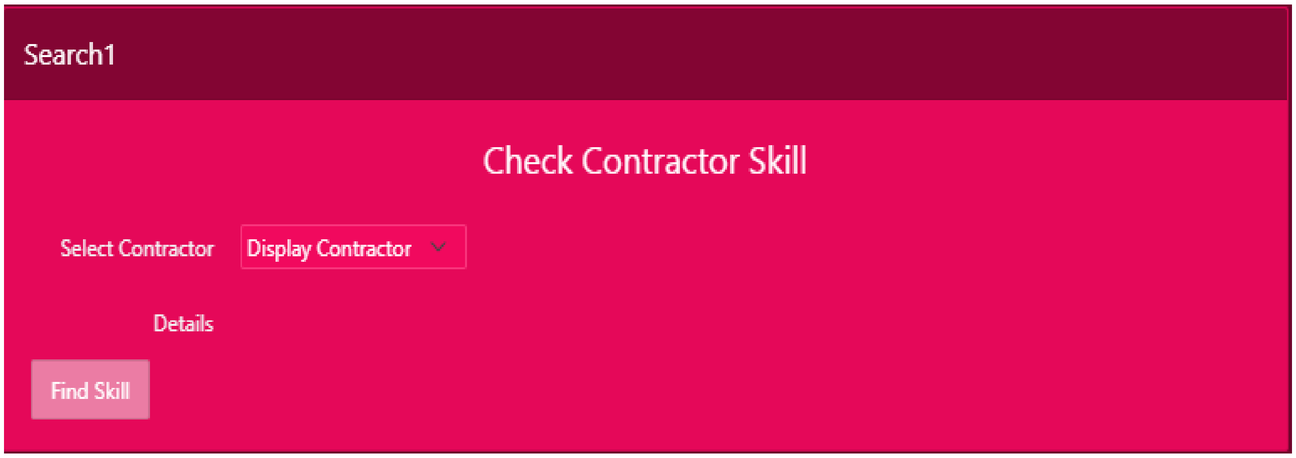
FUNCTION IMPLEMENTATION IN APPLICATION

Function to find contractors Every skill

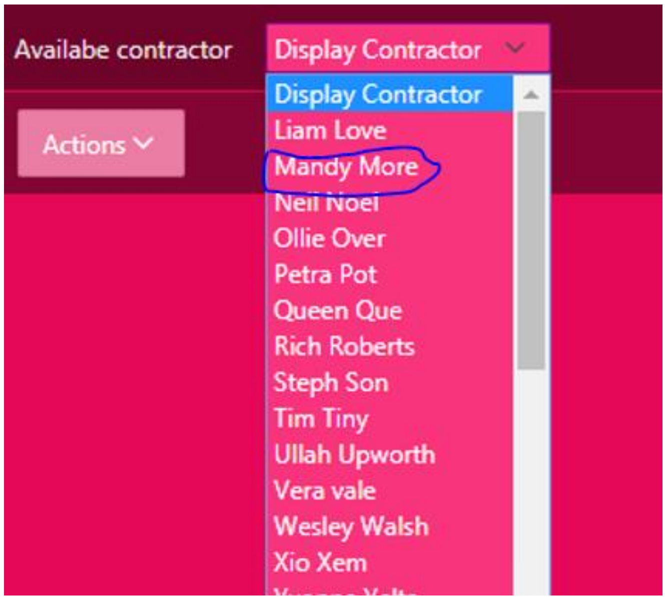


```
begin
select pkg_final.find_con_skill(:P6_NEW) into :P6_NEW_1 from dual;
end;
```

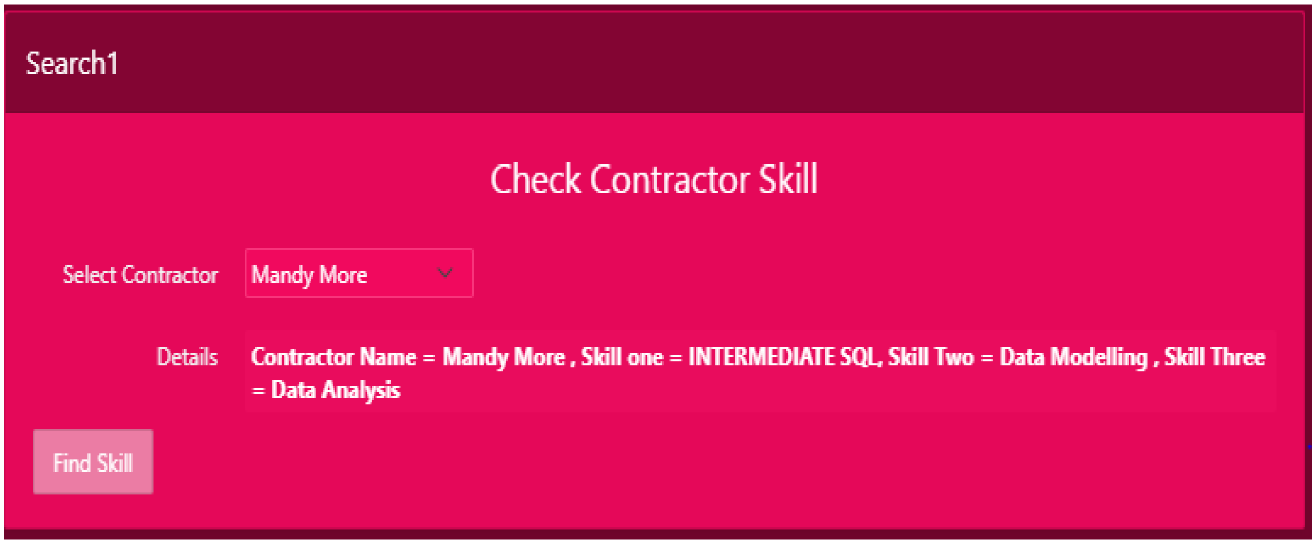
Implementation of function



In order to find contractors skill, we need to select contractor name from the combo box as follows.



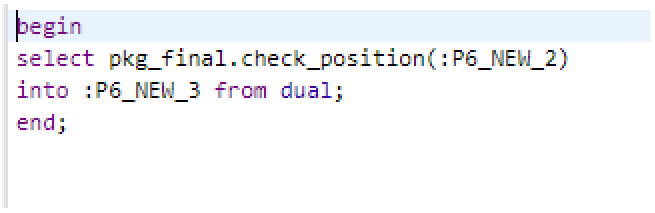
Above Contractor Mandy More is selected from the combo box so Mandy More skill will appear



Details: Contractor Name = Mandy More , Skill one = INTERMEDIATE SQL, Skill Two = Data Modelling , Skill Three = Data Analysis

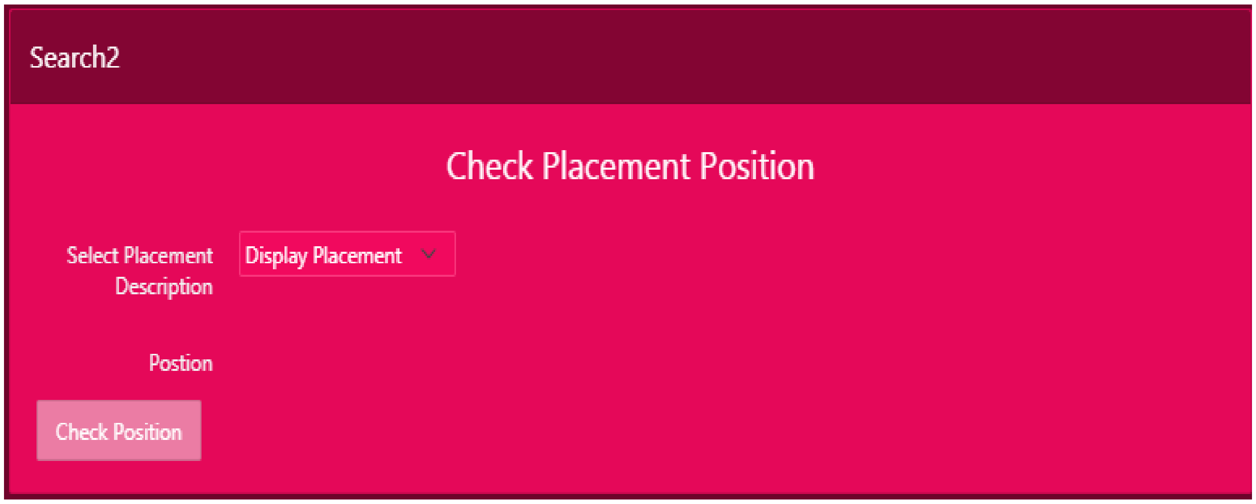
Fig: function find_con_skill work properly

Function to check Placement Position

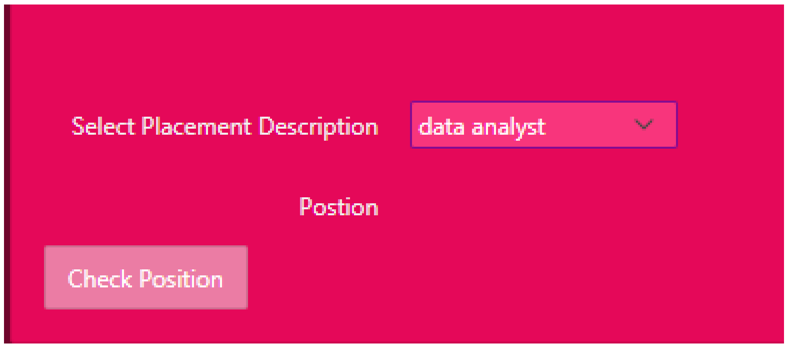


```
begin
select pkg_final.check_position(:P6_NEW_2)
into :P6_NEW_3 from dual;
end;
```

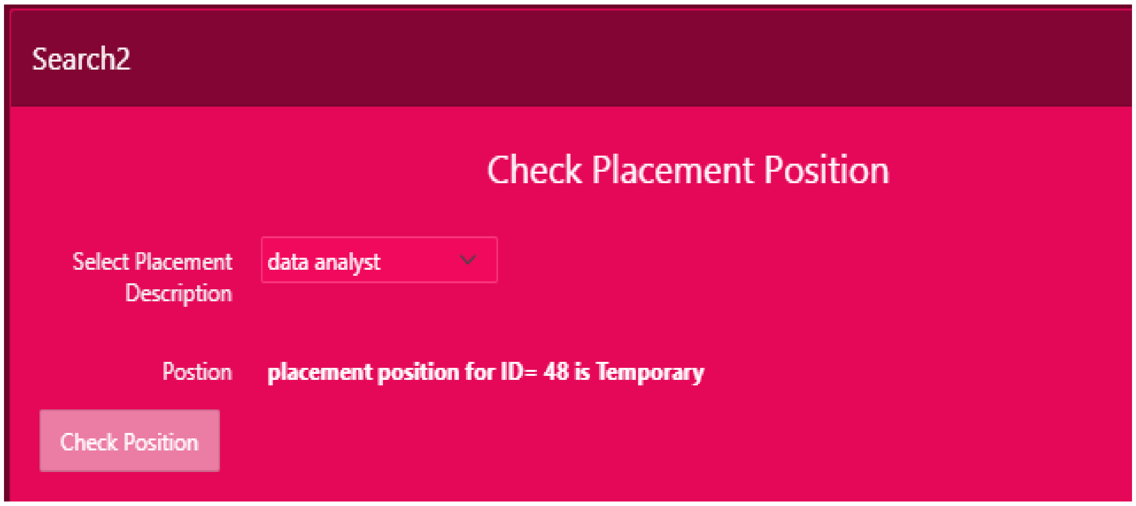
Implementation of function



In order to check placement position, we need to select contractor placement from the combo box as follows.



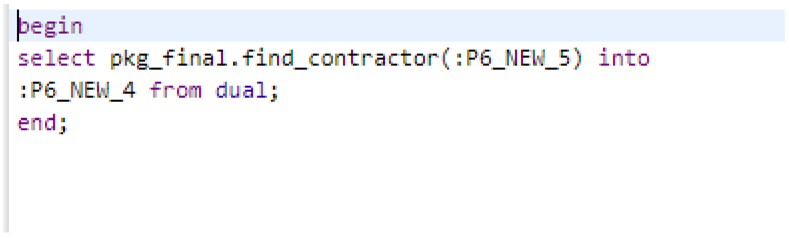
Here placement data analyst is selected to check position



placement position for ID= 48 is Temporary

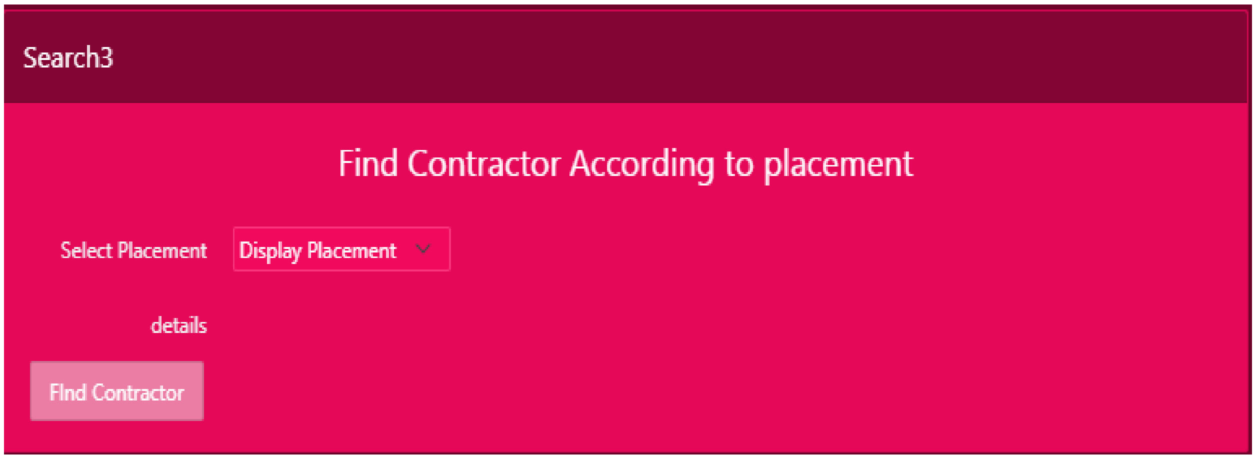
Fig: function check_position work properly

Function to find contractor according to their placement



```
begin
select pkg_final.find_contractor(:P6_NEW_5) into
:P6_NEW_4 from dual;
end;
```

Implementation of function



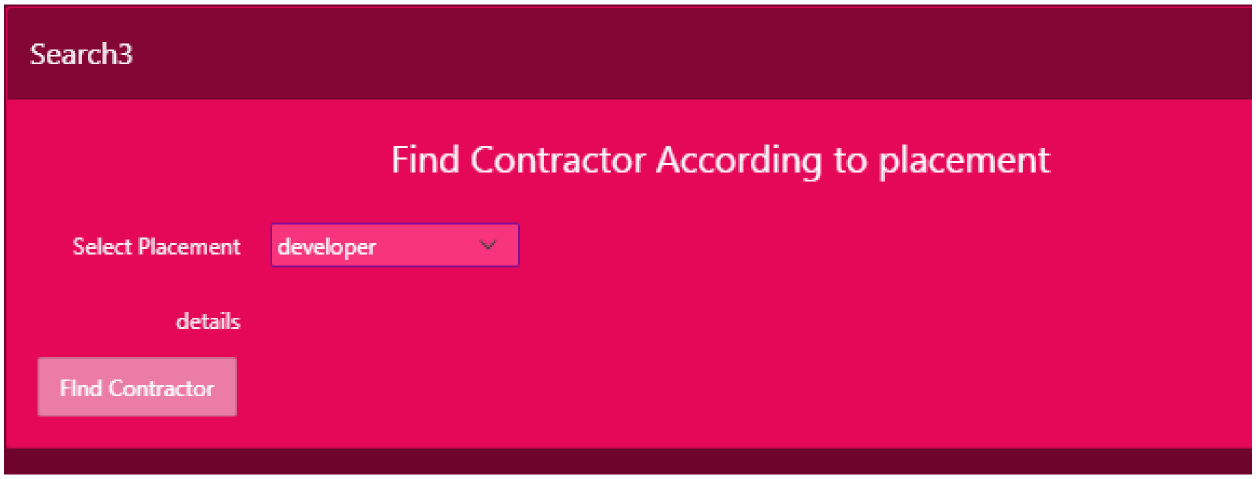
Search3

Find Contractor According to placement

Select Placement: Display Placement

details

In order to find contractor according to their , we need to select contractor placement from the combo box as follows.



Search3

Find Contractor According to placement

Select Placement: developer

details

Here placement developer has been selected from the list, so Contractor who are involved in placement developer should appear.

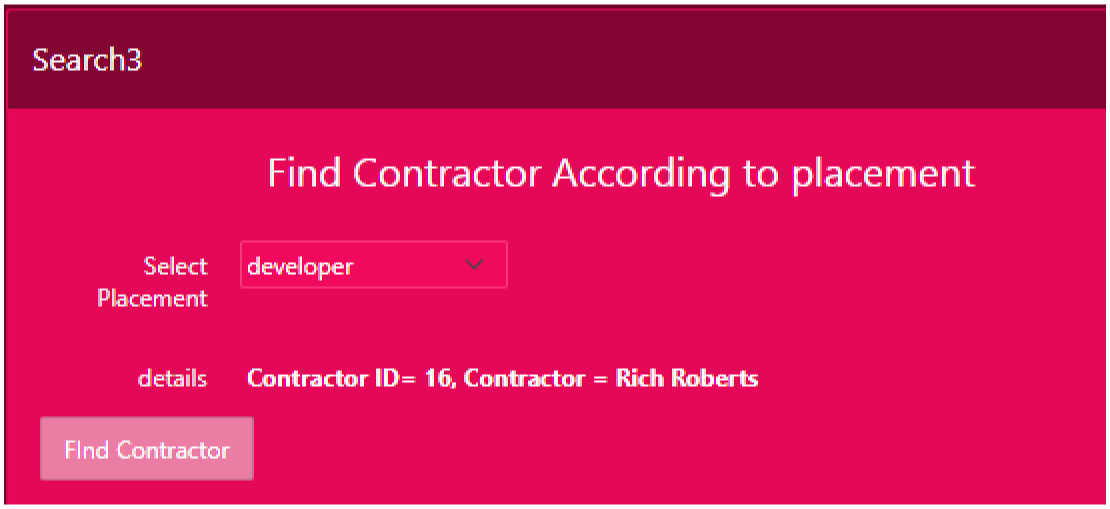
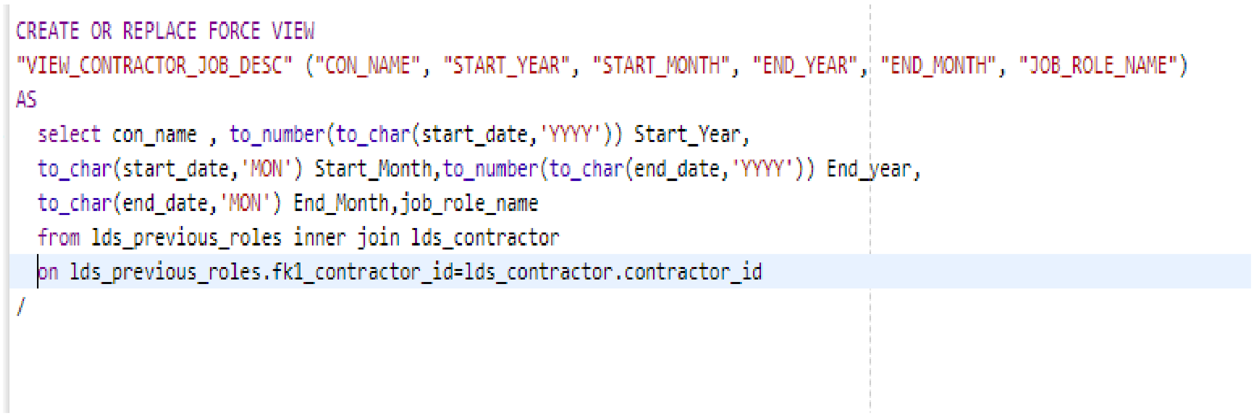

Search3
Find Contractor According to placement
Select Placement
developer
details
Contractor ID= 16, Contractor = Rich Roberts
FInd Contractor

Fig: function find_contractor has worked properly

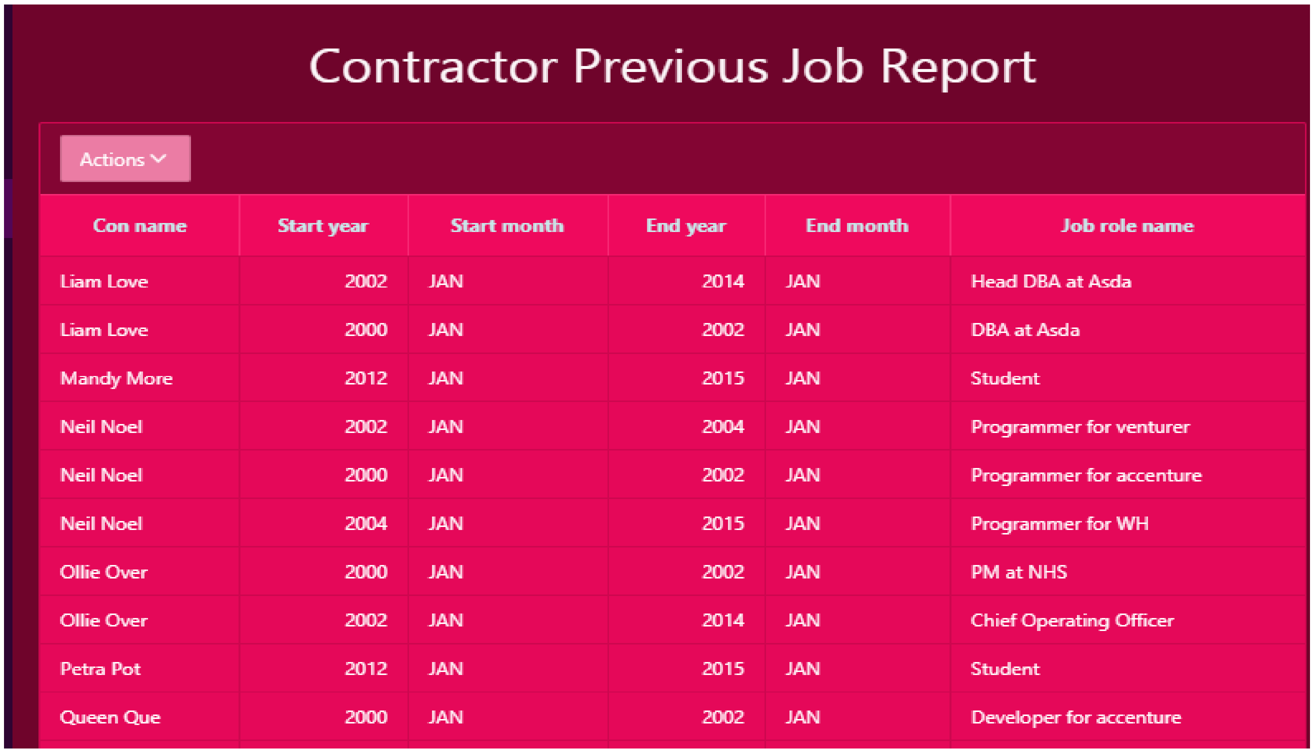
VIEWS IMPLEMENTATION IN THE APPLICATION

Here in this application I have used view to store contractor previous job role details for the report.



```
CREATE OR REPLACE FORCE VIEW
"VIEW_CONTRACTOR_JOB_DESC" ("CON_NAME", "START_YEAR", "START_MONTH", "END_YEAR", "END_MONTH", "JOB_ROLE_NAME")
AS
  select con_name , to_number(to_char(start_date,'YYYY')) Start_Year,
  to_char(start_date,'MON') Start_Month,to_number(to_char(end_date,'YYYY')) End_year,
  to_char(end_date,'MON') End_Month,job_role_name
  from lds_previous_roles inner join lds_contractor
  on lds_previous_roles.fk1_contractor_id=lds_contractor.contractor_id
/
```

Report Generated from is shown below:



Contractor Previous Job Report

Con name	Start year	Start month	End year	End month	Job role name
Liam Love	2002	JAN	2014	JAN	Head DBA at Asda
Liam Love	2000	JAN	2002	JAN	DBA at Asda
Mandy More	2012	JAN	2015	JAN	Student
Neil Noel	2002	JAN	2004	JAN	Programmer for venturer
Neil Noel	2000	JAN	2002	JAN	Programmer for accenture
Neil Noel	2004	JAN	2015	JAN	Programmer for WH
Ollie Over	2000	JAN	2002	JAN	PM at NHS
Ollie Over	2002	JAN	2014	JAN	Chief Operating Officer
Petra Pot	2012	JAN	2015	JAN	Student
Queen Que	2000	JAN	2002	JAN	Developer for accenture

Fig: report of view

PART 3: PERFORMANCE PLAN

INTRODUCTION

A poorly performing database application that impacts transactional speeds and slows down queries can have a significant impact not only on user productivity, but other applications running on the same server or the same network. Underperforming platforms that are not keeping up with the speed of your business puts you at a competitive disadvantage and can ultimately impact the quality of service provided to your end-customer.

WHY PERFORMANCE TUNNING REQUIRED?

Performance Tuning process will identify bottlenecks, diagnose configuration and indexing issues and provide you with a detailed study on how to best optimize your database application and overall computing environment. All aspects of the environment are taken into consideration in order to give you a comprehensive view of the improvements you can make, in the most cost-effective manner. Database Performance Tuning includes:

Architectural Design Review – Evaluation of the quality of the database architecture. Poorly architected databases always perform poorly, and will not see typical improvements from normal tuning. If issues are uncovered, recommendations will be provided for potential redesign

Analysis – Assessment of applications, memory, Disk I/O, OS and overall computing environment

Findings – Identification and recommendations provided for modifications and changes

Implementation – Step-by-step deployment and change management process

QUERY EXPLAIN PLAN

The EXPLAIN PLAN statement displays execution plans chosen by the Oracle optimizer for SELECT, UPDATE, INSERT, and DELETE statements. A statement's execution plan is the sequence of operations Oracle performs to run the statement.

The row source tree is the core of the execution plan. It shows the following information:

- An ordering of the tables referenced by the statement
- An access method for each table mentioned in the statement
- A join method for tables affected by join operations in the statement
- Data operations like filter, sort, or aggregation

In addition to the row source tree, the plan table contains information about the following:

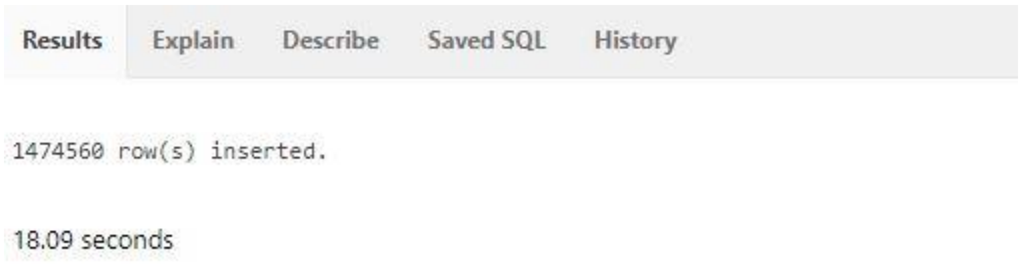
- Optimization, such as the cost and cardinality of each operation
- Partitioning, such as the set of accessed partitions
- Parallel execution, such as the distribution method of join inputs

The EXPLAIN PLAN results let you determine whether the optimizer selects a particular execution plan, such as, nested loops join. It also helps you to understand the optimizer decisions, such as why the optimizer chose a nested loops join instead of a hash join, and lets you understand the performance of a query.

IMPLEMENTATION OF EXECUTION PLAN

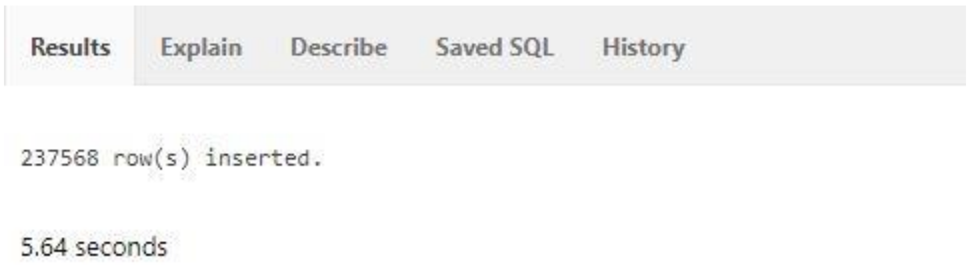
To implements query execution plan I have used certain methodologies which helps to know optimizer decision.

Before doing explain I have done a data grow in two tables lds_contractor_grow and lds_placement_grow . In lds_contractor_grow table I have inserted 1474560 rows and in lds_placement_grow I have inserted 237568 rows.



```
1474560 row(s) inserted.

18.09 seconds
```

fig: data grow in lds_contractor_grow

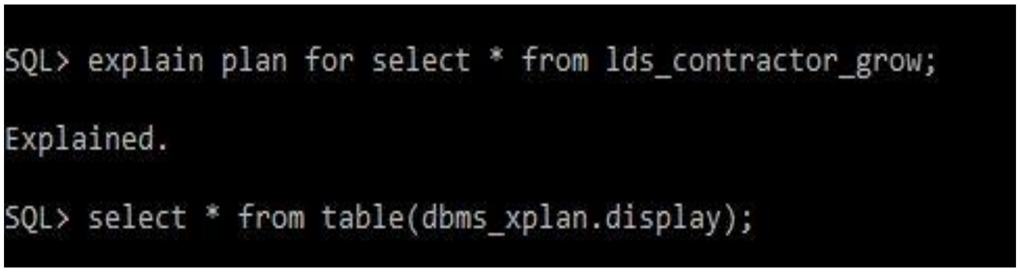


```
237568 row(s) inserted.

5.64 seconds
```

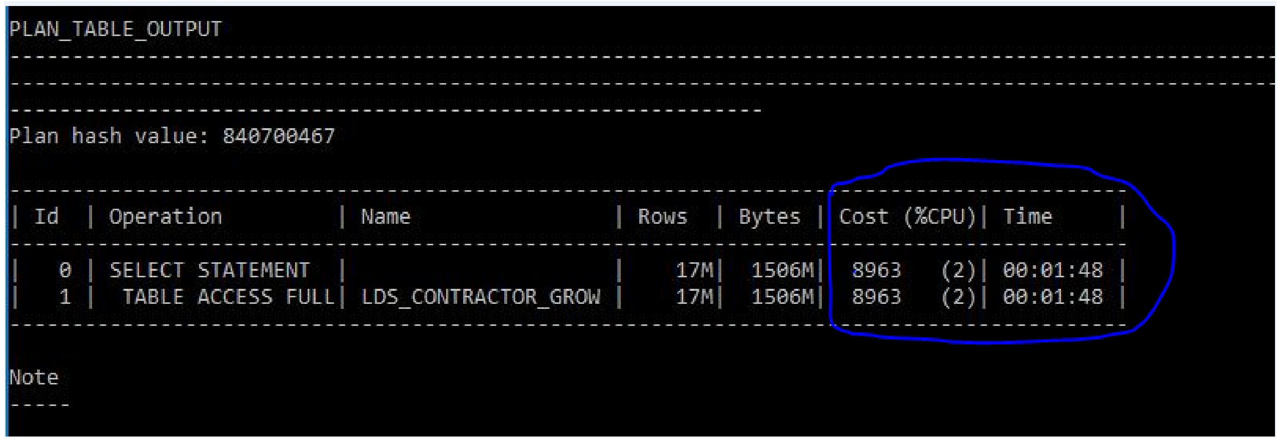
fig: data grow in lds_placement_grow

Comparing Explain Plan with and without primary key

1)Without Primary Key

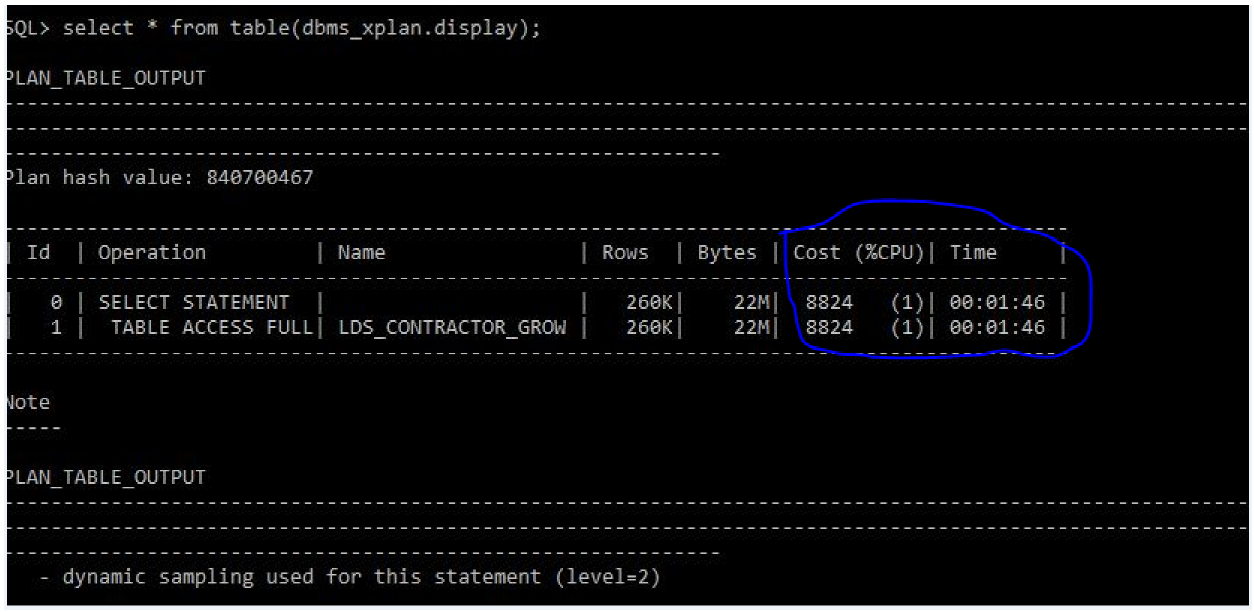


```
SQL> explain plan for select * from lds_contractor_grow;

Explained.

SQL> select * from table(dbms_xplan.display);
```



```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------
Plan hash value: 840700467

-------------------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     |    17M|  1506M|  8963   (2)| 00:01:48 |
|   1 |  TABLE ACCESS FULL| LDS_CONTRACTOR_GROW |    17M|  1506M|  8963   (2)| 00:01:48 |
-------------------------------------------------------------------------------------------

Note
-----
```

2) With Primary Key

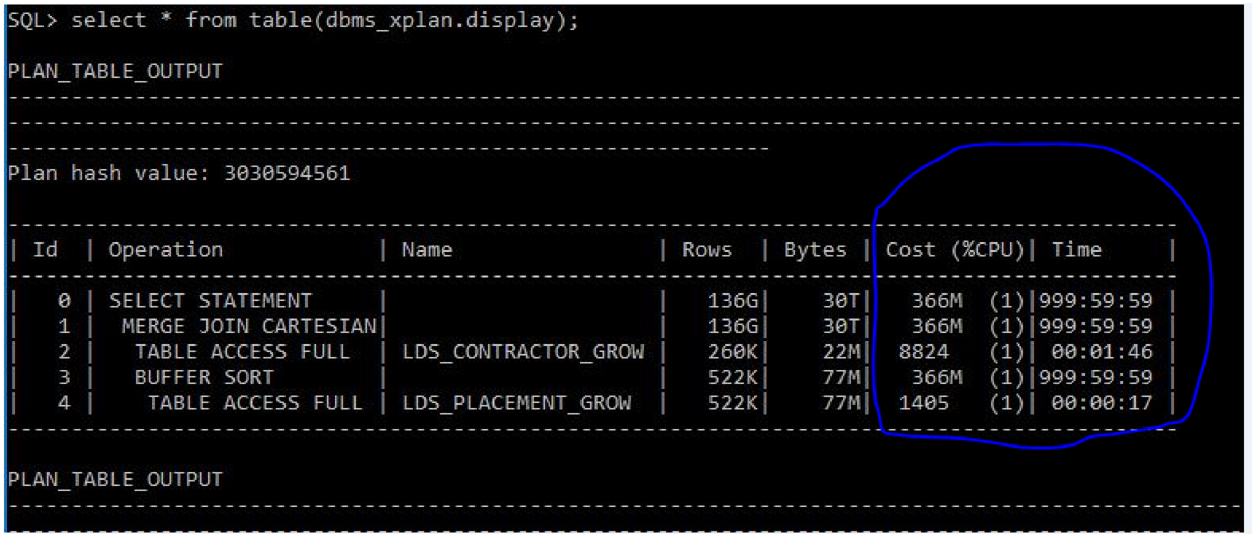


```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------
Plan hash value: 840700467

-------------------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     |   260K|    22M|  8824   (1)| 00:01:46 |
|   1 |  TABLE ACCESS FULL| LDS_CONTRACTOR_GROW |   260K|    22M|  8824   (1)| 00:01:46 |
-------------------------------------------------------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------
   - dynamic sampling used for this statement (level=2)
```

AS compared to without primary key, with primary key query is little fast.

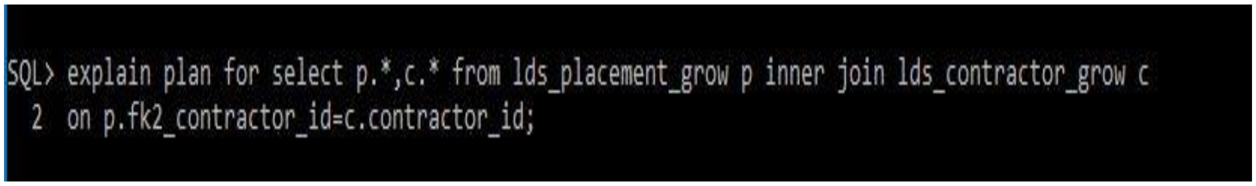
Comparing Cartesian join and inner join

Explain plan of cartesian join

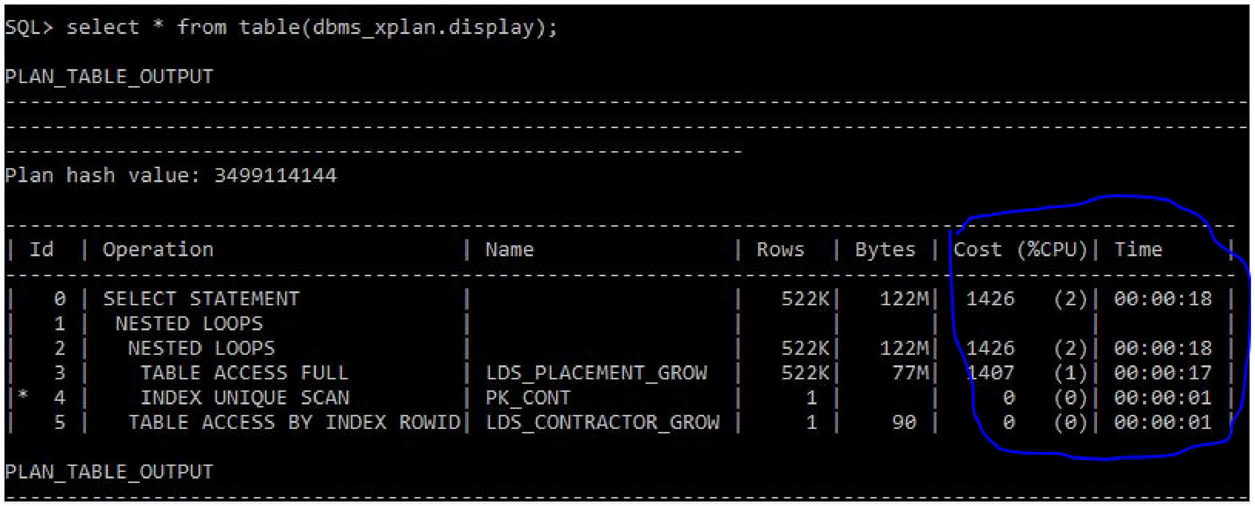


```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------------
Plan hash value: 3030594561

----------------------------------------------------------------------------------------
| Id  | Operation              | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                     |  136G|   30T|   366M  (1)|999:59:59 |
|   1 |  MERGE JOIN CARTESIAN|                     |  136G|   30T|   366M  (1)|999:59:59 |
|   2 |   TABLE ACCESS FULL    | LDS_CONTRACTOR_GROW |  260K|   22M|  8824   (1)| 00:01:46 |
|   3 |   BUFFER SORT          |                     |  522K|   77M|   366M  (1)|999:59:59 |
|   4 |    TABLE ACCESS FULL   | LDS_PLACEMENT_GROW  |  522K|   77M|  1405   (1)| 00:00:17 |
----------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

Explain Plan of inner join



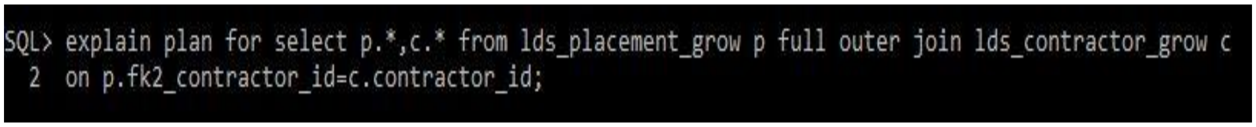
```
SQL> explain plan for select p.*,c.* from lds_placement_grow p inner join lds_contractor_grow c
  2  on p.fk2_contractor_id=c.contractor_id;
```



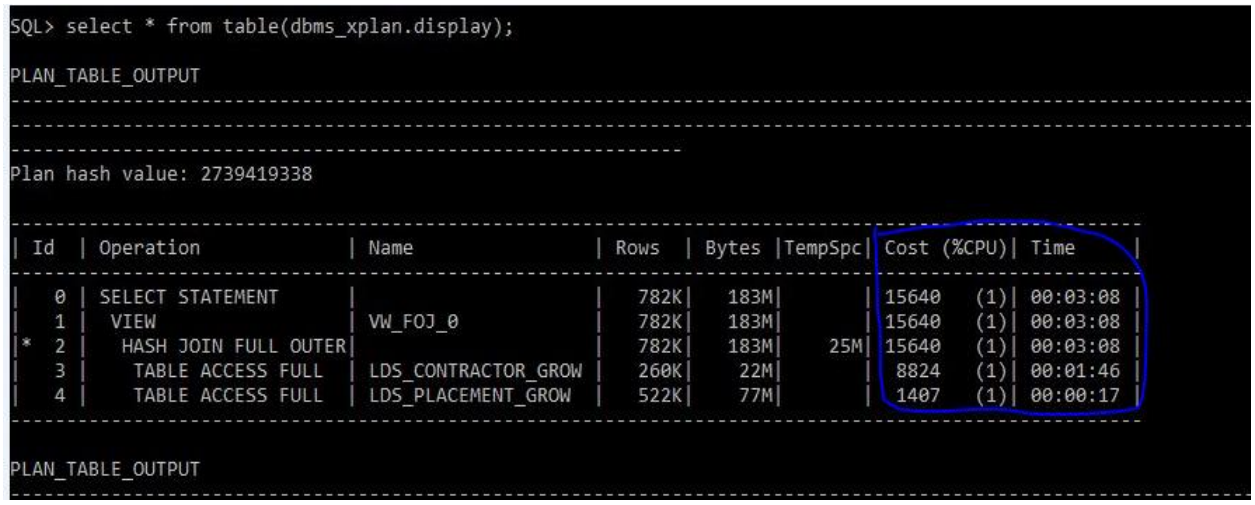
```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------------
Plan hash value: 3499114144

--------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |  522K|  122M|  1426   (2)| 00:00:18 |
|   1 |  NESTED LOOPS                |                     |       |       |            |          |
|   2 |   NESTED LOOPS               |                     |  522K|  122M|  1426   (2)| 00:00:18 |
|   3 |    TABLE ACCESS FULL         | LDS_PLACEMENT_GROW  |  522K|   77M|  1407   (1)| 00:00:17 |
|*  4 |    INDEX UNIQUE SCAN         | PK_CONT             |     1 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| LDS_CONTRACTOR_GROW |     1 |    90 |     0   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
```

As we can see that using inner join is much faster than cartesian join.

Explain plan for Outer join



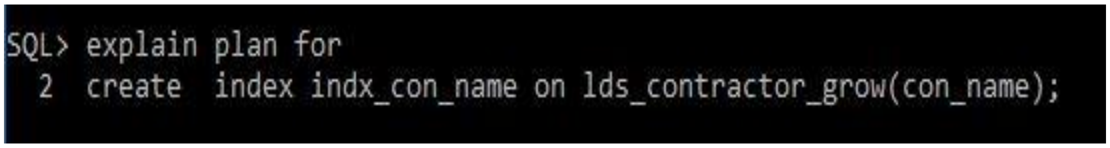
```
SQL> explain plan for select p.*,c.* from lds_placement_grow p full outer join lds_contractor_grow c
  2  on p.fk2_contractor_id=c.contractor_id;
```



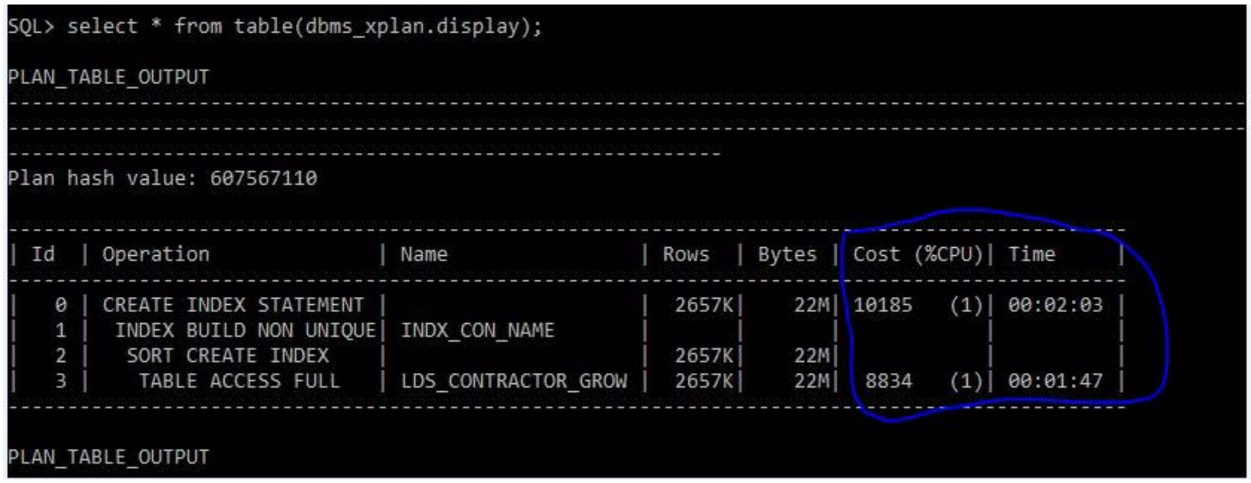
```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
-------------------------------------------------------
Plan hash value: 2739419338

-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    |  782K|   183M|       |15640   (1)| 00:03:08 |
|   1 |  VIEW                 | VW_FOJ_0           |  782K|   183M|       |15640   (1)| 00:03:08 |
|*  2 |   HASH JOIN FULL OUTER|                    |  782K|   183M|    25M|15640   (1)| 00:03:08 |
|   3 |    TABLE ACCESS FULL  | LDS_CONTRACTOR_GROW |  260K|    22M|       | 8824   (1)| 00:01:46 |
|   4 |    TABLE ACCESS FULL  | LDS_PLACEMENT_GROW  |  522K|    77M|       | 1407   (1)| 00:00:17 |
-------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
```

Outer join is little slower than inner join and faster than Cartesian join.

Explain plan using index



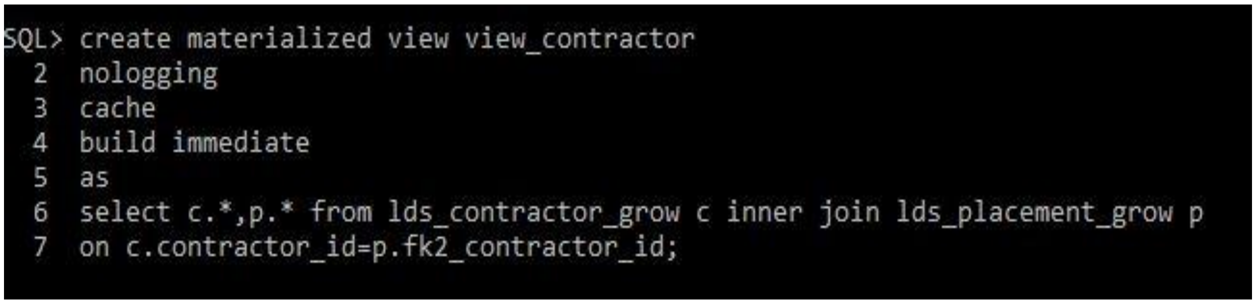
```
SQL> explain plan for
  2  create  index indx_con_name on lds_contractor_grow(con_name);
```



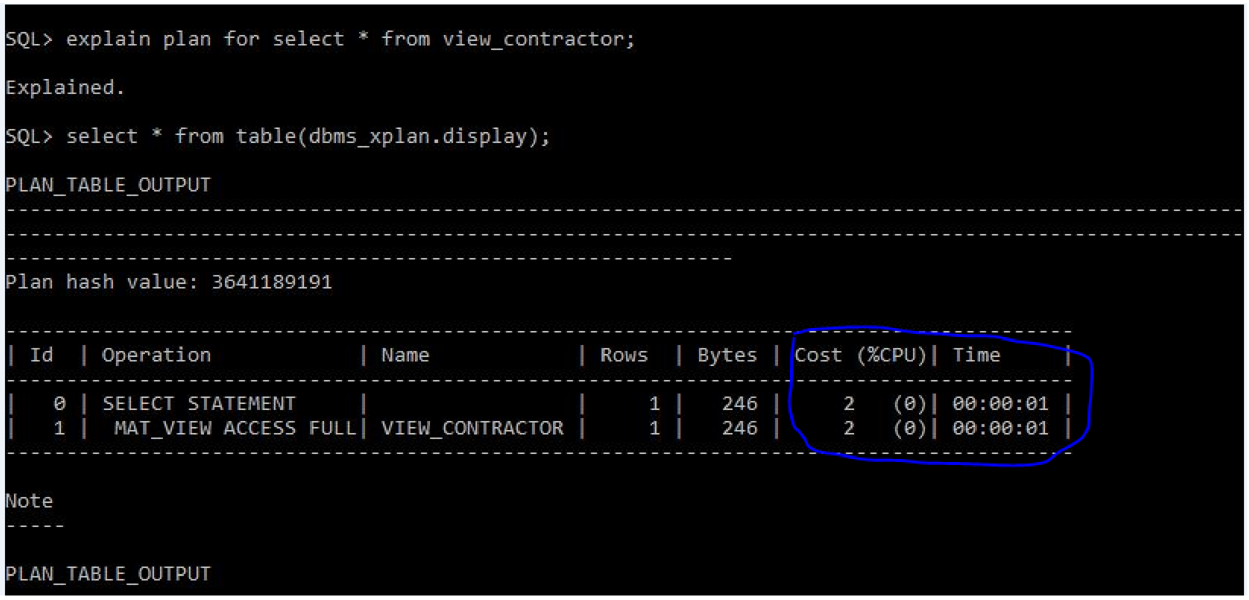
```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
----------------------------------------------------------
Plan hash value: 607567110

----------------------------------------------------------------------------------------------
| Id  | Operation              | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | CREATE INDEX STATEMENT |                     |  2657K|   22M| 10185   (1)| 00:02:03 |
|   1 |  INDEX BUILD NON UNIQUE| INDX_CON_NAME       |       |      |            |          |
|   2 |   SORT CREATE INDEX    |                     |  2657K|   22M|            |          |
|   3 |    TABLE ACCESS FULL   | LDS_CONTRACTOR_GROW |  2657K|   22M|  8834   (1)| 00:01:47 |
----------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
```

Using index helps our query to fetch data faster comparing to non-index column.

Explain plan using Materialized view



```
SQL> create materialized view view_contractor
  2  nologging
  3  cache
  4  build immediate
  5  as
  6  select c.*,p.* from lds_contractor_grow c inner join lds_placement_grow p
  7  on c.contractor_id=p.fk2_contractor_id;
```



```
SQL> explain plan for select * from view_contractor;

Explained.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
------------------------------------------------------------
Plan hash value: 3641189191

-----------------------------------------------------------------------------------------
| Id  | Operation             | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                  |     1 |   246 |     2   (0)| 00:00:01 |
|   1 |  MAT_VIEW ACCESS FULL| VIEW_CONTRACTOR |     1 |   246 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
```

Comparing to all others using view made our query much more faster.