**Lecture 3**

**What is PL/SQL?**

* PL/SQL Stands for Procedural Language extension to SQL
* It is platform independence because we can run it on many platforms

**Why PLSQL**

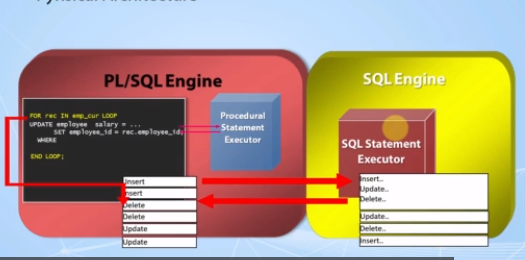
* PL/SQL is not a display language.You cannot create visual things with PL/SQL
* It is developed for business operations or handling some complicated logic with strong logic control statements
* Besides,you can do some after sql statements like logs,controls like constraints,some security things, and reusability with the stored procedures.
* PL/SQL works in database.So It will be much faster than the other languages in its area
* PL/SQL is just created for only data manipulation and there are built in functions which we cannot find in other programming language
* PLSQL is created just for data manipulation

**Lecture 4: PL/SQL Architecture**

* There are two architectures of PL/SQL.one is physical architecture

**Physical Architecture**

* When we run query it steps into three main processes.Parsing,Fetching and Executing.This operation is done by SQL engine.
* When you run your query SQL optimizer optimizes your query.

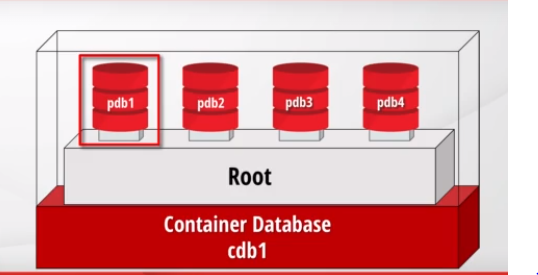


**Logical Architecture**

* **Cooperates with SQL Engine**
  + PL/SQL engine allows us to create, manage and execute SQL and PL/SQL codes and interact with the database
  + If you write an SQL code inside your PL/SQL codes, it will call the SQL engine and have this operation done with SQL engine. Then the result is sent back to the PL/SQL engine. This operation is called as context switches .One engine to another, and the opposite is valid.
* **Enables SubPrograms**
* **Dynamic Queries**
* **Case Insensitivity**
* **Optimizer**
* **Enables Object-Oriented Programming**
* **Web development**

**Lecture 5:What is Pluggable Databases?**

Pluggable database =Multitenant Architecture



* Container databases does not have it’s objects,tables,etc.CDB store only metadata such as configuration files,etc.
* Pluggable databases have it’s own objects,tables,etc.
* In earlier versions of database,each database should be installed on a separate server.But in companies,there are very small databases used for only some specific jobs.So there is no reason to dedicate another server for such a small database.Besides,each server is a lot of work for the DBAs.To make that easier,oracle updated its architecture now,there are pluggable databases in one container database.

**Lecture 6:The Sample(HR) schema used in this Course**

**What is Schema ?**

* Schemas are the collections of objects for each user in oracle database.

**What is HR Schema?**

* Abbreviation of Human Resources. A Schema that can practice on it.
* A schema can have objects like;
* Tables
* Views
* Triggers
* Constraints

**Lecture 8 : About the Database Installation**

**There are two ways to install database:**

**1.Using virtual box**

**2.using install directly database in a laptop**

**Lecture 9 : The Ways to Get a Database**

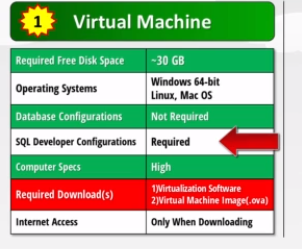
We offer 3 options to work with a database

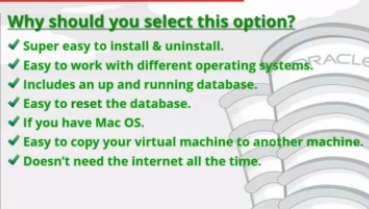
1.Virtual Machine

2.Local Installation

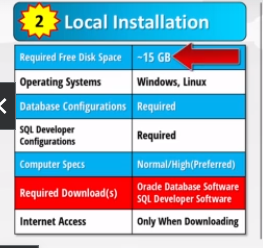
3.Oracle Live SQL

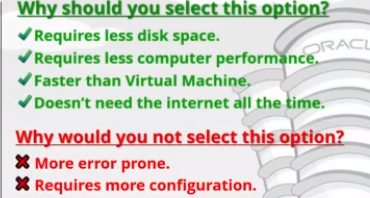
**1.Virtual Machine🡪** To go with this option,we need to install VMware software



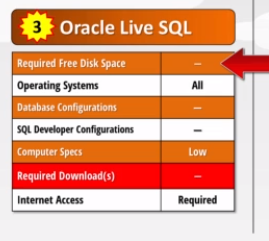


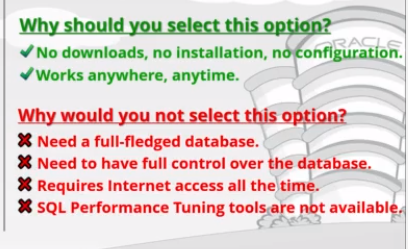
**2.Local Installation**





**3.Using Oracle Live SQL**





**Lecture 23 : Using Oracle Live SQL**

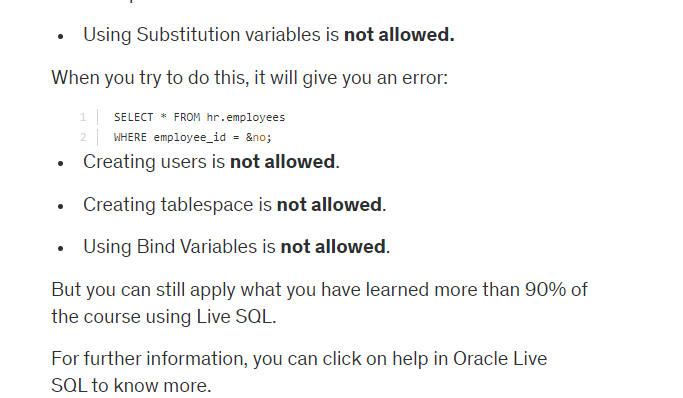
**Search Oracle LIVE SQL in a browser**

[**https://livesql.oracle.com/apex/f?p=590:1000 (Go**](https://livesql.oracle.com/apex/f?p=590:1000%20(Go) **to this link) and click on start coding button**

**username –** [**shalineemotarwar@gmail.com**](mailto:shalineemotarwar@gmail.com)

**password – Sanket@14**

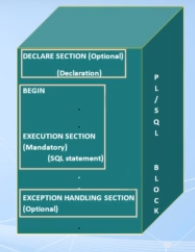
**Lecture 24 :About Oracle Live SQL Restrictions**



**Lecture 25 :Anonymous Blocks**

1. **Blocks--🡪**

* In PL/SQL programming,you need to write all your code in blocks.
* Blocks are the executable code groups that are inside the “begin end”key words.
* Declare section(optional)
* Begin Section is mandatory
* Exception section is optional
* END;(mandatory)



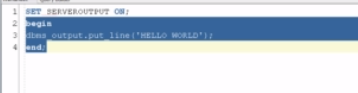
**Create HR schema in SQL developer**

There are 3 types of block

* Anonymous Blocks
* Procedures---🡪it doesn’t return a value
* Functions--🡪it must return a value

**Lecture 26 :PL/SQL Outputs**

1. **Not an output language**
2. **SET SERVEROUTPUT ON**
3. **DBMS\_OUTPUT -🡪** It is an oracle’s prebuilt package that can do some output operations



SET Serveroutput on is needed in Oracle SQL developer software

**4.Nested Blocks**

<https://livesql.oracle.com/apex/livesql/s/1e478gzeym455pcj7rbfupg7>

**Lecture 27: What are variables and Why to use them?**



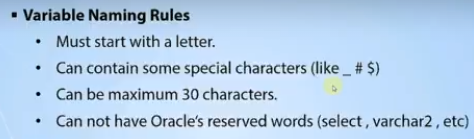
**1.Reference Datatype--🡪This datatypes hold value which point to a storage location**

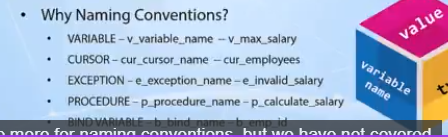
**2.Large Objects---🡪This are also pointers to other data items that are stored outside of table such as images,files,etc.**

**3.Composite-🡪collections(this datatype can hold more than one value in it).**

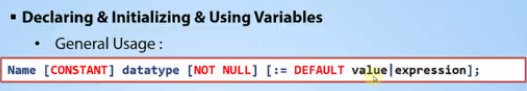


**Lecture 28 : Naming Rules and Naming Conventions**





**Lecture 29:Declaring and Initializing & Using Variables Part1**



<https://livesql.oracle.com/apex/livesql/s/1e52pzqo7nqemn7o1nq3lkkq>



**Lecture 30 : Declaring and Initializing & Using Variables Part2**

Declare

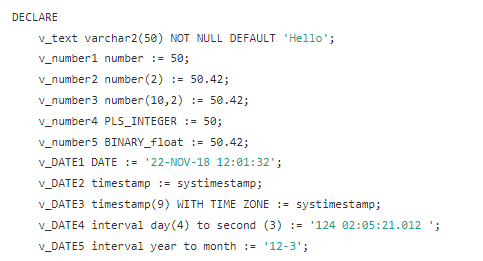
V\_number PLS\_INTEGER(10,5) NOT NULL :=50.42

BEGIN

DBMS\_OUTPUT.put\_line(V\_number || ‘shalu’);

END;

**Lecture 31 : Declaring and Initializing & Using Variables (Code Samples)**



**Lecture 32 : Using % Attribute**

The %TYPE attribute is used to declare variables according to the already declared variable or database column. It is used when you are declaring an individual variable, not a record. The data type and precision of the variable declared using %TYPE attribute is the same as that of the column that is referred from a given table. This is particularly useful when declaring variables that will hold database values. When using the %TYPE keyword, the name of the columns and the table to which the variable will correspond must be known to the user. These are then prefixed with the variable name. If some previously declared variable is referred then prefix that variable name to the %TYPE attribute.

**Syntax:** <var\_name> <tab\_name>.<column\_name>%TYPE;

**Example:**

declare

v\_type emp.name%type;

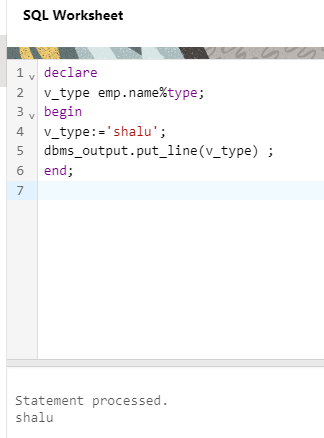
begin

v\_type:='shalu';

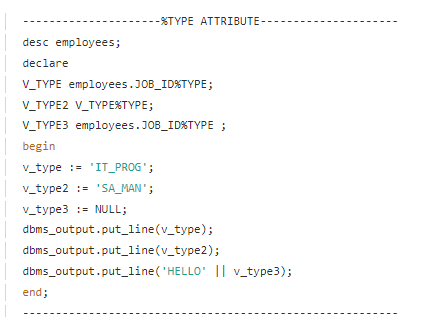
dbms\_output.put\_line(v\_type) ;

end;

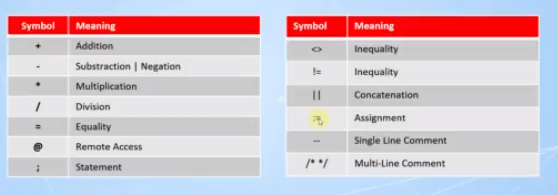
<https://livesql.oracle.com/apex/livesql/s/1iiu4xf1o8ew8qq22vziczi6>



**Lecture 33 : Using %Type Attribute (Code Samples)**

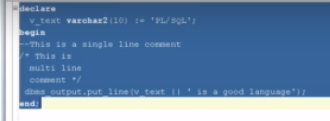


**Lecture 34 :PL/SQL Delimiters and Commenting Your Code**

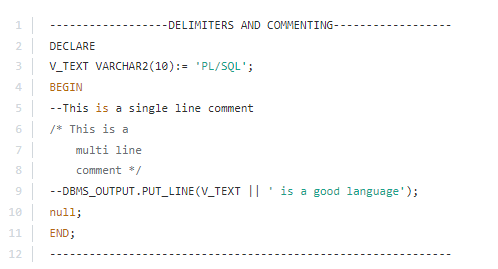


**---(single line comment)**

**/\* \*/ -🡪Multi line comment**



**Lecture 35 :PL/SQL Delimiters and Commenting Your Code(Code Samples)**



**Lecture 36 :PL/SQL Variable Scope**

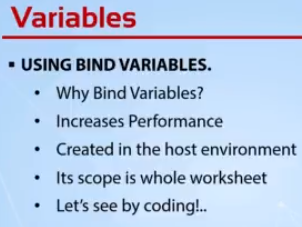
1. begin <<outer>>
2. DECLARE
3. --v\_outer VARCHAR2(50) := 'Outer Variable!';
4. v\_text VARCHAR2(20) := 'Out-text';
5. BEGIN
6. DECLARE
7. v\_text VARCHAR2(20) := 'In-text';
8. v\_inner VARCHAR2(30) := 'Inner Variable';
9. BEGIN
10. --dbms\_output.put\_line('inside -> ' || v\_outer);
11. --dbms\_output.put\_line('inside -> ' || v\_inner);
12. dbms\_output.put\_line('inner -> ' || v\_text);
13. dbms\_output.put\_line('outer -> ' || outer.v\_text);
14. END;
15. --dbms\_output.put\_line('inside -> ' || v\_inner);
16. --dbms\_output.put\_line(v\_outer);
17. dbms\_output.put\_line(v\_text);
18. END;
19. END outer;

<https://livesql.oracle.com/apex/livesql/s/1izv1v0iomtyb7muw5gsevim>

**Lecture 37 :PL/SQL Variable Scope(Code Samples)**



**Lecture 38: Using Bind Variables**



**-🡪We can access bind variables from another block where bind variable is not declared.**

**-🡪We need to call the bind variables explicitly,it’s automatically called during run and it will print the value;**

**-🡪We cannot directly assign the values to a bind variable**

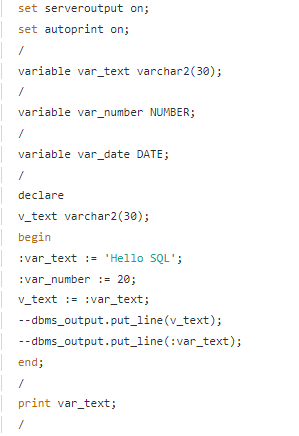
**Syntax for declaration of bind variable:**

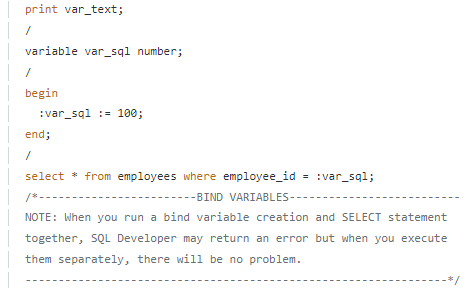
Variable var\_sql number;////var\_sql is the name of bind variable

**Syntax for definition of bind variable:**

var\_sql=40;

1. set serveroutput on;
2. set autoprint on;
3. /
4. variable var\_text varchar2(30);
5. /
6. variable var\_number NUMBER;
7. /
8. variable var\_date DATE;
9. /
10. declare
11. v\_text varchar2(30);
12. begin
13. :var\_text := 'Hello SQL';
14. :var\_number := 20;
15. v\_text := :var\_text;
16. --dbms\_output.put\_line(v\_text);
17. --dbms\_output.put\_line(:var\_text);
18. end;
19. /
20. print var\_text;
21. /
22. variable var\_sql number;
23. /
24. begin
25. :var\_sql := 100;
26. end;
27. /
28. select \* from employees where employee\_id = :var\_sql;
29. /\*------------------------BIND VARIABLES--------------------------
30. NOTE: When you run a bind variable creation and SELECT statement
31. together, SQL Developer may return an error but when you execute
32. them separately, there will be no problem.
33. **Lecture 39: Using Bind Variables (Code Samples)**





**Lecture 40: What Are Control Structures and If Statements**

**Lecture 41: What Are Control Structures and If Statements ( Code Samples)**

**Lecture 42:Case Expressions**

**Lecture 43:Case Expressions(Code Samples)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The Complete PL/SQL Bootcamp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author : Oracle Master Training \*

\* Course : The Complete Oracle PL/SQL Bootcamp \*

\* Lecture : CASE Expressions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Simple Case Expression \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_job\_code VARCHAR2(10) := 'SA\_MAN';

v\_salary\_increase NUMBER;

BEGIN

v\_salary\_increase := CASE v\_job\_code

WHEN 'SA\_MAN' THEN 0.2

WHEN 'SA\_REP' THEN 0.3

ELSE 0

END;

dbms\_output.put\_line('Your salary increase is : '|| v\_salary\_increase);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Searched Case Expression \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_job\_code VARCHAR2(10) := 'IT\_PROG';

v\_department VARCHAR2(10) := 'IT';

v\_salary\_increase NUMBER;

BEGIN

v\_salary\_increase:=CASE

WHEN v\_job\_code = 'SA\_MAN' THEN 0.2

WHEN v\_department = 'IT' AND v\_job\_code = 'IT\_PROG' THEN 0.3

ELSE 0

END;

dbms\_output.put\_line('Your salary increase is : '|| v\_salary\_increase);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CASE Statements \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_job\_code VARCHAR2(10) := 'IT\_PROG';

v\_department VARCHAR2(10) := 'IT';

v\_salary\_increase NUMBER;

BEGIN

CASE

WHEN v\_job\_code = 'SA\_MAN' THEN

v\_salary\_increase := 0.2;

dbms\_output.put\_line('The salary increase for a Sales Manager is: '|| v\_salary\_increase);

WHEN v\_department = 'IT' AND v\_job\_code = 'IT\_PROG' THEN

v\_salary\_increase := 0.2;

dbms\_output.put\_line('The salary increase for a Sales Manager is: '|| v\_salary\_increase);

ELSE

v\_salary\_increase := 0;

dbms\_output.put\_line('The salary increase for this job code is: '|| v\_salary\_increase);

END CASE;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Lecture 44: What Are Basic Loops**

**Lecture 45: What Are Basic Loops (Code Samples)**

DECLARE

v\_counter NUMBER(2) := 1;

BEGIN

LOOP

dbms\_output.put\_line('My counter is : '|| v\_counter);

v\_counter := v\_counter + 1;

--IF v\_counter = 10 THEN

-- dbms\_output.put\_line('Now I reached : '|| v\_counter);

-- EXIT;

--END IF;

EXIT WHEN v\_counter > 10;

END LOOP;

END;

**Lecture 46: While Loops** 

**Lecture 47: While Loops (Code Samples)**

DECLARE

v\_counter NUMBER(2) := 1;

BEGIN

WHILE v\_counter <= 10 LOOP

dbms\_output.put\_line('My counter is : '|| v\_counter);

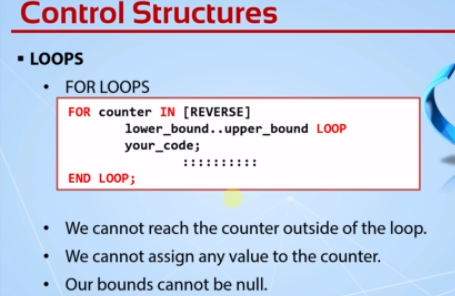
v\_counter := v\_counter + 1;

-- EXIT WHEN v\_counter > 3;

END LOOP;

END;

**Lecture 48: For Loops**



**Lecture 49: For Loops(Code Samples)**

BEGIN

FOR i IN REVERSE 1..3 LOOP

dbms\_output.put\_line('My counter is : ' || i);

END LOOP;

END;

**Lecture 50: Nested Loops & Loop Labeling**

**Lecture 51: Nested Loops & Loop Labeling(Code Samples)**

DECLARE

v\_inner NUMBER := 1;

BEGIN

FOR v\_outer IN 1..5 LOOP

dbms\_output.put\_line('My outer value is : ' || v\_outer );

v\_inner := 1;

LOOP

v\_inner := v\_inner+1;

dbms\_output.put\_line(' My inner value is : ' || v\_inner );

EXIT WHEN v\_inner \* v\_outer >= 15;

END LOOP;

END LOOP;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Nested Loops with Labels \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_inner NUMBER := 1;

BEGIN

<<outer\_loop>>

FOR v\_outer IN 1..5 LOOP

dbms\_output.put\_line('My outer value is : ' || v\_outer );

v\_inner := 1;

<<inner\_loop>>

LOOP

v\_inner := v\_inner+1;

dbms\_output.put\_line(' My inner value is : ' || v\_inner );

EXIT outer\_loop WHEN v\_inner \* v\_outer >= 16;

EXIT WHEN v\_inner \* v\_outer >= 15;

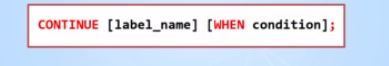
END LOOP inner\_loop;

END LOOP outer\_loop;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Lecture 52: Continue Statement**



**Lecture 53: Continue Statement(Code Samples)**

DECLARE

v\_inner NUMBER := 1;

BEGIN

FOR v\_outer IN 1..10 LOOP

dbms\_output.put\_line('My outer value is : ' || v\_outer );

v\_inner := 1;

WHILE v\_inner \* v\_outer < 15 LOOP

v\_inner := v\_inner + 1;

CONTINUE WHEN MOD(v\_inner \* v\_outer,3) = 0;

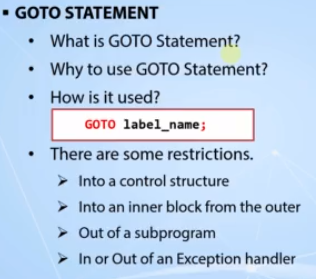
dbms\_output.put\_line(' My inner value is : ' || v\_inner );

END LOOP;

END LOOP;

END;

**Lecture 54: GOTO Statement**



**Lecture 54: GOTO Statement(Code Samples)**

DECLARE

v\_searched\_number NUMBER := 22;

v\_is\_prime BOOLEAN := true;

BEGIN

FOR x IN 2..v\_searched\_number-1 LOOP

IF v\_searched\_number MOD x = 0 THEN

dbms\_output.put\_line(v\_searched\_number|| ' is not a prime number..');

v\_is\_prime := false;

GOTO end\_point;

END IF;

END LOOP;

IF v\_is\_prime THEN

dbms\_output.put\_line(v\_searched\_number|| ' is a prime number..');

END IF;

<<end\_point>>

dbms\_output.put\_line('Check complete..');

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_searched\_number NUMBER := 32457;

v\_is\_prime BOOLEAN := TRUE;

x NUMBER := 2;

BEGIN

<<start\_point>>

IF v\_searched\_number MOD x = 0 THEN

dbms\_output.put\_line(v\_searched\_number|| ' is not a prime number..');

v\_is\_prime := FALSE;

GOTO end\_point;

END IF;

x := x+1;

IF x = v\_searched\_number THEN

GOTO prime\_point;

END IF;

GOTO start\_point;

<<prime\_point>>

IF v\_is\_prime THEN

dbms\_output.put\_line(v\_searched\_number || ' is a prime number..');

END IF;

<<end\_point>>

dbms\_output.put\_line('Check complete..');

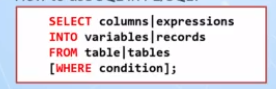
END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Lecture 56: Operating with Selected Database data**

Transactions ends with a commit,rollback or a system failure.

**How to use SQL in PL/SQL?**



**Lecture 57: Operating with Selected Database data(Code Samples)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_name VARCHAR2(50);

v\_salary employees.salary%type;

BEGIN

SELECT first\_name ||' '|| last\_name, salary

INTO v\_name, v\_salary

FROM employees

WHERE employee\_id = 130;

dbms\_output.put\_line('The salary of '|| v\_name || ' is : '|| v\_salary);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_name VARCHAR2(50);

sysdates employees.hire\_date%type;

BEGIN

SELECT first\_name ||' '|| last\_name, sysdates

INTO v\_name, sysdates

FROM employees

WHERE employee\_id = 130;

dbms\_output.put\_line('The salary of '|| v\_name || ' is : '|| sysdates);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_name VARCHAR2(50);

v\_sysdate employees.hire\_date%type;

employee\_id employees.employee\_id%type := 130;

BEGIN

SELECT first\_name ||' '|| last\_name, sysdate

INTO v\_name, v\_sysdate

FROM employees

WHERE employee\_id = employee\_id;

dbms\_output.put\_line('The salary of '|| v\_name || ' is : '|| v\_sysdate );

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_name VARCHAR2(50);

v\_salary employees.salary%type;

v\_employee\_id employees.employee\_id%type := 130;

BEGIN

SELECT first\_name ||' '|| last\_name, salary

INTO v\_name, v\_salary

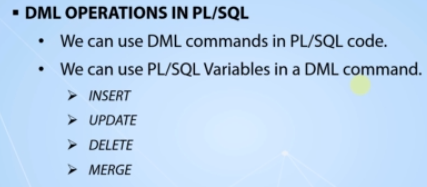
FROM employees

WHERE employee\_id = v\_employee\_id;

dbms\_output.put\_line('The salary of '|| v\_name || ' is : '|| v\_salary );

END;

**Lecture 58: DML Operations in PL/SQL**



**Lecture 59: DML Operations in PL/SQL(Code Samples)**

**CREATE TABLE employees\_copy**

AS SELECT \* FROM employees;

DECLARE

v\_employee\_id PLS\_INTEGER := 0;

v\_salary\_increase NUMBER := 400;

BEGIN

FOR i IN 217..226 LOOP

-- INSERT INTO employees\_copy

-- (employee\_id, first\_name, last\_name, email, hire\_date, job\_id, salary)

-- VALUES

-- (i, 'employee#'||i,'temp\_emp','abc@xmail.com',sysdate,'IT\_PROG',1000);

-- UPDATE employees\_copy

-- SET salary = salary + v\_salary\_increase

-- WHERE employee\_id = i;

DELETE FROM employees\_copy

WHERE employee\_id = i;

END LOOP;

END;

**Lecture 60: Using Sequences in PL/SQL**

CREATE SEQUENCE employee\_id\_seq

START WITH 207

INCREMENT BY 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BEGIN

FOR i IN 1..10 LOOP

INSERT INTO employees\_copy

(employee\_id,first\_name,last\_name,email,hire\_date,job\_id,salary)

VALUES

(employee\_id\_seq.nextval,'employee#'||employee\_id\_seq.nextval,'temp\_emp','abc@xmail.com',sysdate,'IT\_PROG',1000);

END LOOP;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_seq\_num NUMBER;

BEGIN

SELECT employee\_id\_seq.nextval

INTO v\_seq\_num

FROM dual;

dbms\_output.put\_line(v\_seq\_num);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_seq\_num NUMBER;

BEGIN

SELECT employee\_id\_seq.nextval

INTO v\_seq\_num

FROM employees\_copy

WHERE rownum = 1;

dbms\_output.put\_line(v\_seq\_num);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

v\_seq\_num NUMBER;

BEGIN

v\_seq\_num := employee\_id\_seq.nextval;

dbms\_output.put\_line(v\_seq\_num);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BEGIN

dbms\_output.put\_line(employee\_id\_seq.nextval);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

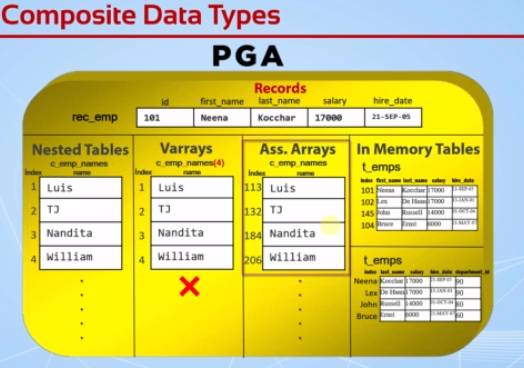
BEGIN

dbms\_output.put\_line(employee\_id\_seq.currval);

END;

**Lecture 62: Extra – Enabling Server Outputs by Default**

**Lecture 63: Simple Data Types vs Composite Data Types**



**Nested Tables-🡪**We can add indexes dynamically according to the need.We don’t need to declare it’s size.and the indexes starts from 1.

**Varrays🡪** We can’ add indexes dynamically according to the need.We need to declare it’s size.and the indexes starts from 1

**Ass.Arrays🡪** We can add indexes dynamically according to the need.We don’t need to declare it’s size.and the indexes starts from any number and indexes can be a negative number also.

**Collections-🡪** Nested Tables,Varrays,Ass.Arrays,In Memory Tables

**Lecture 64:PL/SQL Records – Part1**

**We can store a whole row in a record.All the variables stored in a record are related to each other.We can add one record inside a another record.By default all the variables in a record are null.**



**We get all of the column’s types of a row into one record in one step. Using %rowtype**

**Syntax:**

**Way 1: record\_name table\_name%rowtype;**

**Example:**

DECLARE

r\_emp employees%rowtype;

BEGIN

SELECT \* INTO r\_emp

FROM employees

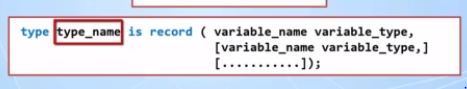
WHERE employee\_id = '101';

--r\_emp.salary := 2000;

dbms\_output.put\_line(r\_emp.first\_name || ' ' || r\_emp.last\_name || ' earns '|| r\_emp.salary || ' and hired at : ' || r\_emp.hire\_date);

end;

**Way 2:**



**Example:**

DECLARE

--r\_emp employees%rowtype;

type t\_emp IS RECORD (first\_name VARCHAR2(50), last\_name employees.last\_name%TYPE, salary employees.salary%TYPE, hire\_date DATE);

r\_emp t\_emp;

BEGIN

SELECT first\_name,last\_name,salary,hire\_date INTO r\_emp

FROM employees WHERE employee\_id = '101';

dbms\_output.put\_line(r\_emp.first\_name || ' ' || r\_emp.last\_name || ' earns ' || r\_emp.salary || ' and hired at : ' || r\_emp.hire\_date);

END;



Every Record can get only one level.

**Lecture 65:PL/SQL Records – Part2**

DECLARE

TYPE t\_edu is RECORD(primary\_school VARCHAR2(100),

high\_school VARCHAR2(100),

university VARCHAR2(100),

uni\_graduate\_date DATE

);

TYPE t\_emp IS RECORD(first\_name VARCHAR2(50),

last\_name employees.last\_name%type,

salary employees.salary%type NOT NULL DEFAULT 1000,

hire\_date DATE,

dept\_id employees.department\_id%type,

department departments%rowtype,

education t\_edu

);

r\_emp t\_emp;

BEGIN

SELECT first\_name, last\_name, salary, hire\_date, department\_id

INTO r\_emp.first\_name, r\_emp.last\_name, r\_emp.salary, r\_emp.hire\_date, r\_emp.dept\_id

FROM employees where employee\_id = '146';

SELECT \*

INTO r\_emp.department

FROM departments

WHERE department\_id = r\_emp.dept\_id;

r\_emp.education.high\_school := 'Beverly Hills';

r\_emp.education.university := 'Oxford';

r\_emp.education.uni\_graduate\_date := '01-JAN-13';

dbms\_output.put\_line(r\_emp.first\_name || ' ' ||

r\_emp.last\_name || ' earns ' ||

r\_emp.salary || ' and hired at : ' ||

r\_emp.hire\_date);

dbms\_output.put\_line('She graduated from ' ||

r\_emp.education.university ||

' at ' ||

r\_emp.education.uni\_graduate\_date);

dbms\_output.put\_line('Her Department Name is : '|| r\_emp.department.department\_name);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/