ADS

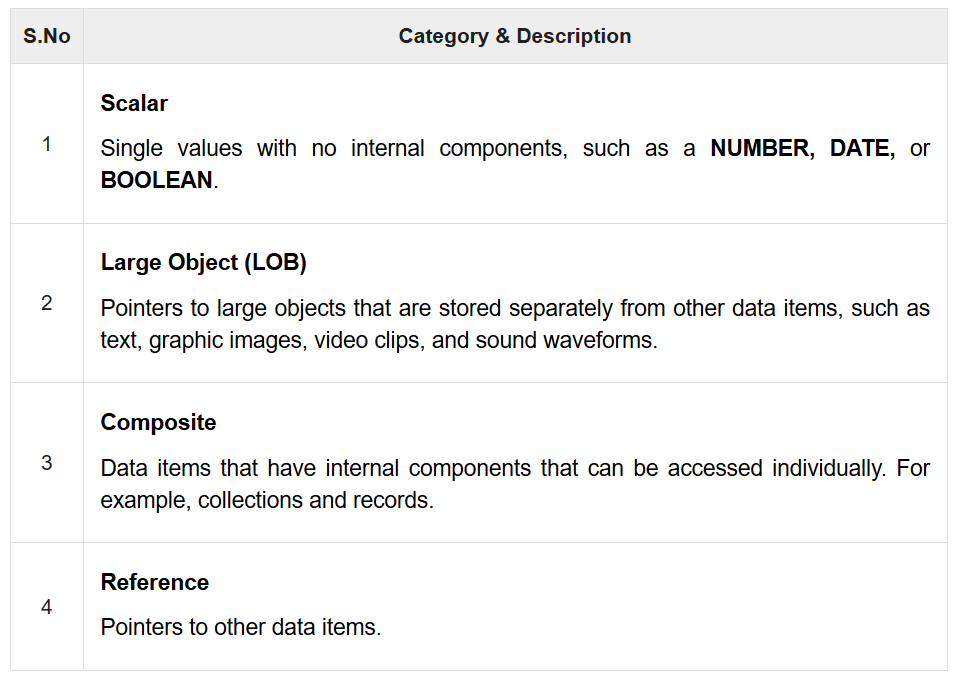
PL/SQL:

1. Developed by oracle corporation in late 1980’s.
2. Stands for “Procedural Language extension of SQL”.
3. Procedural extension language for SQL and oracle relational database.
4. Portable and provides high performance transaction processing language.
5. Block structured language that can have multiple blocks in it.

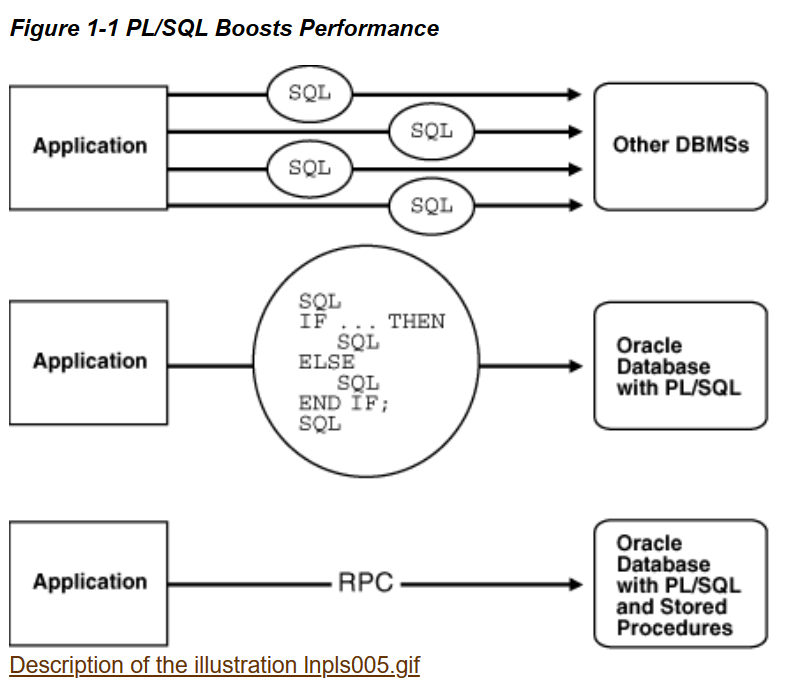
<https://www.guru99.com/introduction-pl-sql.html>

Features of PL/SQL (ADVANTAGES):

1. Tightly integrated with SQL: You do not have to translate between SQL and PL/SQL datatypes; a NUMBER or VARCHAR2 column in the database is stored in a NUMBER or VARCHAR2 variable in PL/SQL. This integration saves you both learning time and processing time. Special PL/SQL language features let you work with table columns and rows without specifying the datatypes, saving on maintenance work when the table definitions change.
2. Offers numerous data types:



1. Offers extensive error checking: It can be done using exceptions.
2. Supports object-oriented programming: Object types are an ideal object-oriented modelling tool, which you can use to reduce the cost and time required to build complex applications. Besides allowing you to create software components that are modular, maintainable, and reusable, object types allow different teams of programmers to develop software components concurrently.
3. Supports structured programming through functions and procedures.
4. Supports the development of web applications and server pages: With PL/SQL you can create applications that generate Web pages directly from an Oracle database. PL/SQL Server Pages (PSPs) enable you to develop Web pages with dynamic content. They are an alternative to coding a stored procedure that writes out the HTML code for a web page, one line at a time.
5. Offers a variety of programming structures.
6. Provides better performance: Without PL/SQL, Oracle must process SQL statements one at a time. Programs that issue many SQL statements require multiple calls to the database, resulting in significant network and performance overhead. With PL/SQL, an entire block of statements can be sent to Oracle at one time.



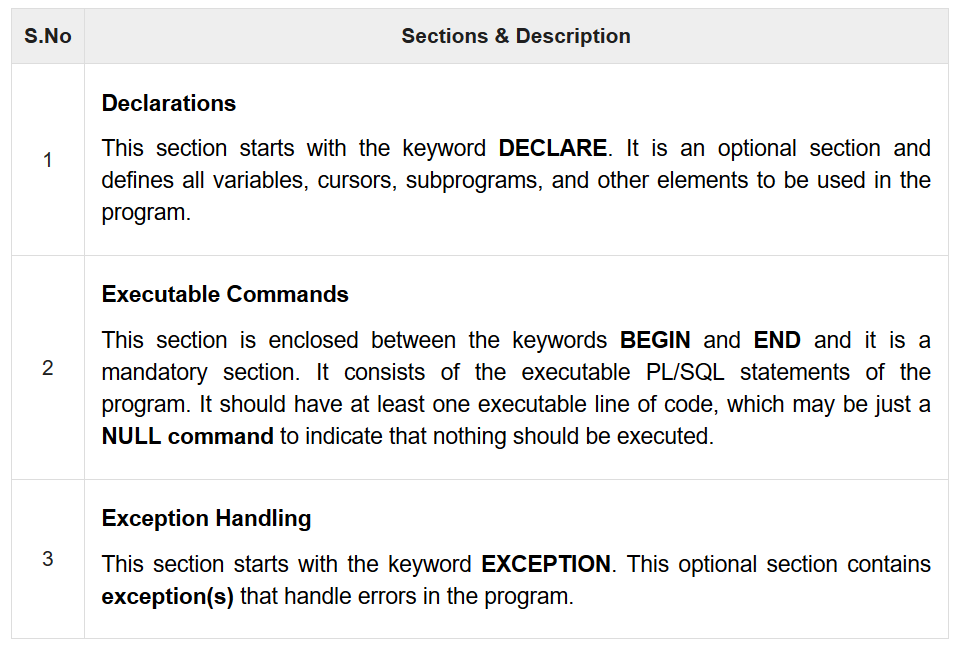
1. Full portability: Applications written in PL/SQL can run on any operating system and platform where the Oracle database runs. With PL/SQL, you can write portable program libraries and reuse them in different environments.
2. Higher productivity: PL/SQL lets you write very compact code for manipulating data. PL/SQL saves time on design and debugging by offering a full range of software-engineering features, such as exception handling, encapsulation, data hiding, and object-oriented datatypes.
3. PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.

Environment:

PL/SQL is not a standalone programming language; it is a tool within the Oracle programming environment. SQL\* Plus is an interactive tool that allows you to type SQL and PL/SQL statements at the command prompt. These commands are then sent to the database for processing. Once the statements are processed, the results are sent back and displayed on screen.

Basic syntax

PL/SQL which is a block-structured language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts –



1. Every PL/SQL statement ends with a semicolon (;).
2. PL/SQL blocks can be nested within other PL/SQL blocks using BEGIN and END.
3. Following is the basic structure of a PL/SQL block –

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling>

END;

1. The end; line signals the end of the PL/SQL block.
2. To run the code from the SQL command line, you may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the result.

Identifiers

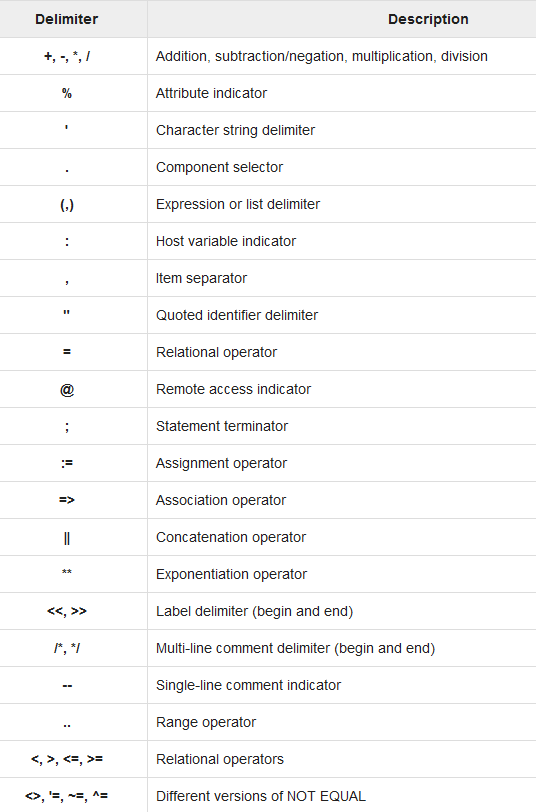
PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words.

How identifiers are declared?

1. Start with a letter.
2. Can be followed by more letters, numerals, dollar sign, underscores and number signs.
3. Should not exceed 30 characters.

Delimiter

A delimiter is a symbol with a special meaning.



Comments

Program comments are explanatory statements that can be included in the PL/SQL code that you write and helps anyone reading its source code.

The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler.

1. Single-line comments start with the delimiter -- (double hyphen).
2. Multi-line comments are enclosed by /\* and \*/.

DECLARE

-- variable declaration

message varchar2(20):= 'Hello, World!';

BEGIN

/\*

\* PL/SQL executable statement(s)

\*/

dbms\_output.put\_line(message);

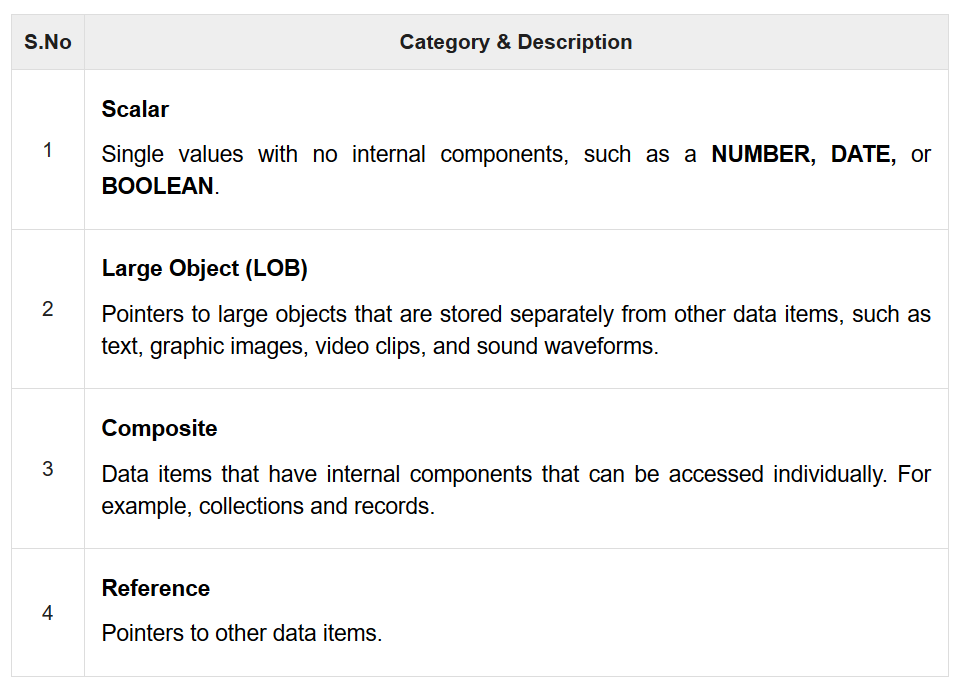
END;

Data types

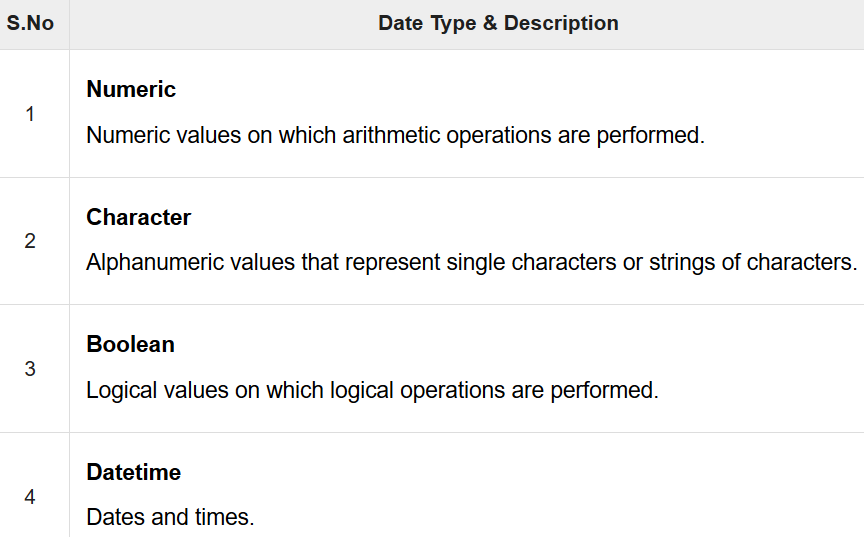
The PL/SQL variables, constants and parameters must have a valid data type, which specifies :

1. a storage format
2. constraints
3. a valid range of values

Following are categories of the datatypes available in PL/SQL:



**Scalar Data Types and Subtypes**

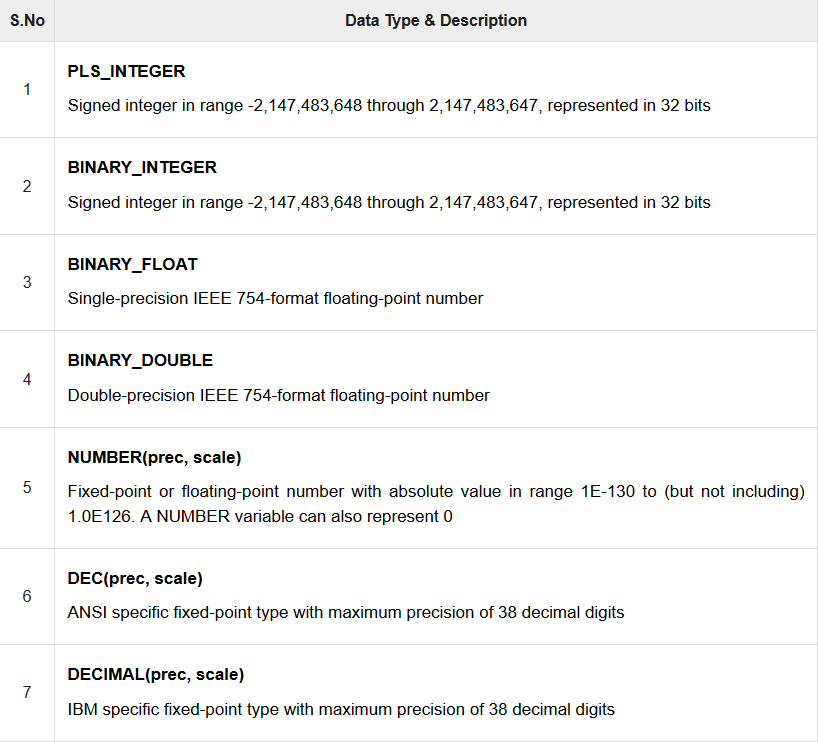


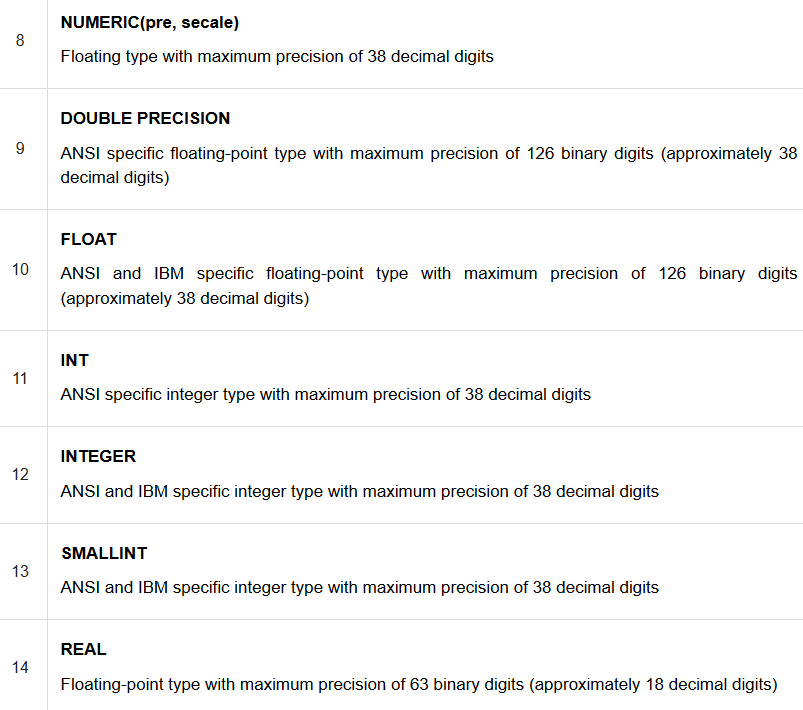
Numeric:

PL/SQL provides subtypes of data types.

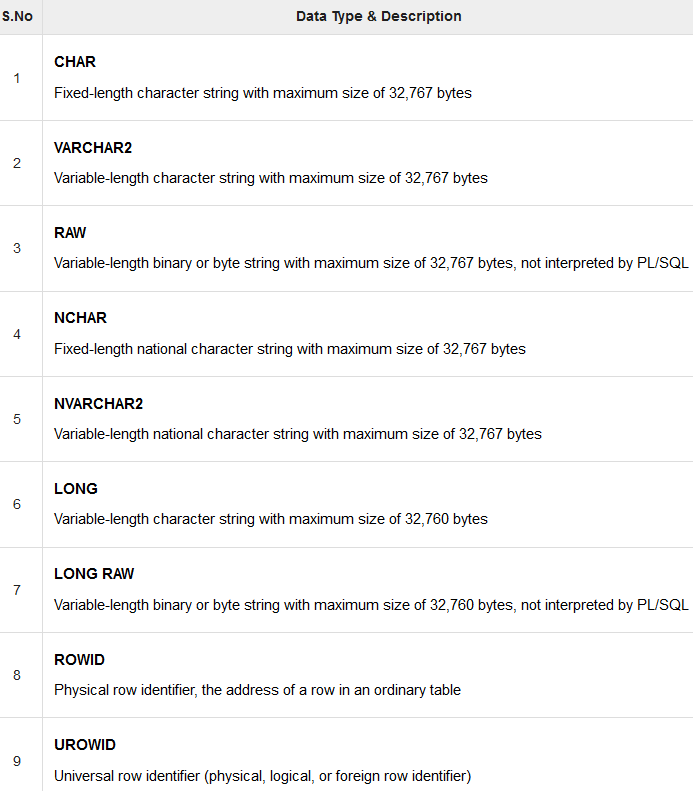
For example, the data type NUMBER has a subtype called INTEGER.

You can use the subtypes in your PL/SQL program to make the data types compatible with data types in other programs while embedding the PL/SQL code in another program, such as a Java program.





Character



Boolean

Stores logical values that are used in logical operations.

The logical values are the Boolean values **TRUE** and **FALSE** and the value **NULL**.

However, SQL has no data type equivalent to BOOLEAN.

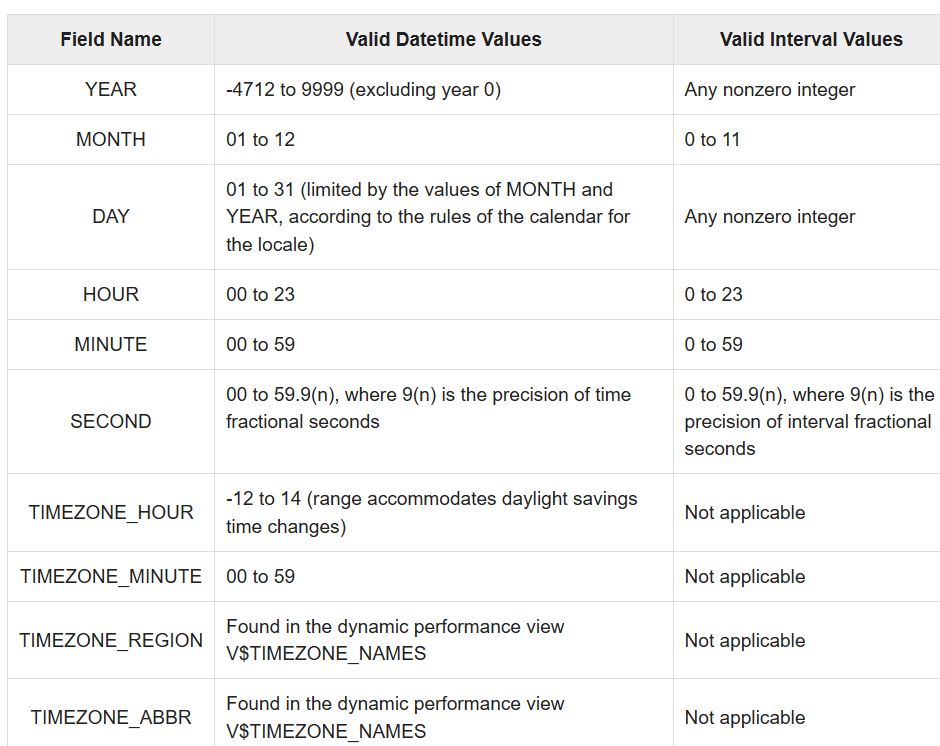
Therefore, Boolean values cannot be used in −

* SQL statements
* Built-in SQL functions (such as **TO\_CHAR**)
* PL/SQL functions invoked from SQL statements

Date time and interval datatypes

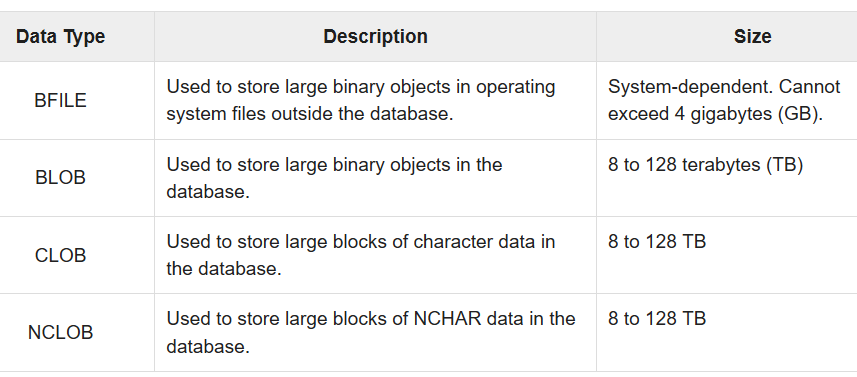
The DATE datatype is used to store fixed-length datetimes, which include the time of day in seconds since midnight. Valid dates range from January 1, 4712 BC to December 31, 9999 AD.

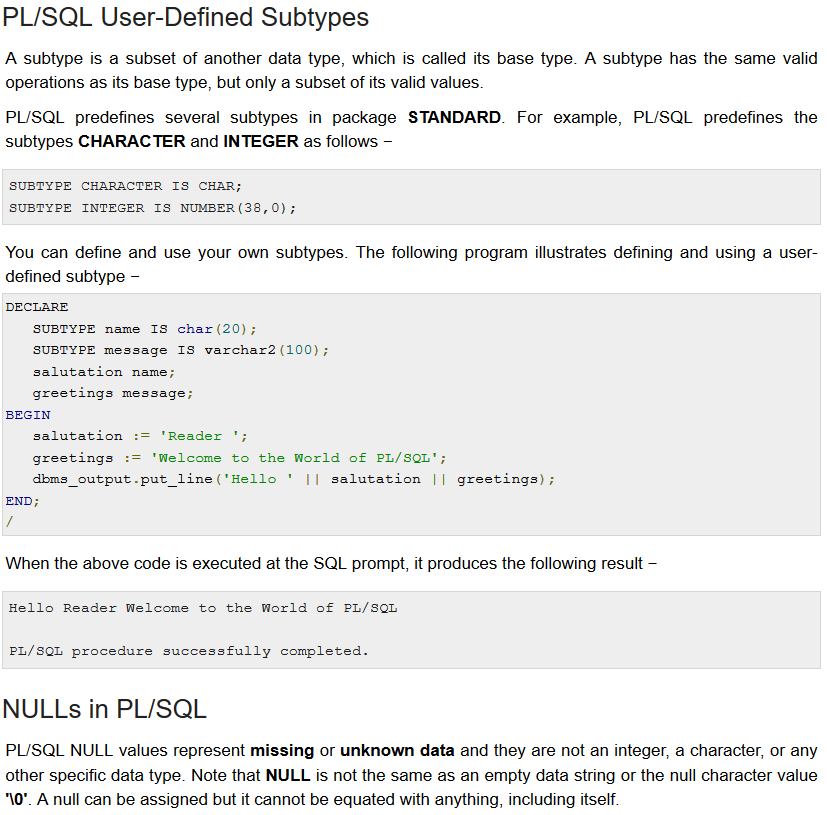
The default date format is set by the Oracle initialization parameter NLS\_DATE\_FORMAT. For example, the default might be 'DD-MON-YY', which includes a two-digit number for the day of the month, an abbreviation of the month name, and the last two digits of the year. For example, 01-OCT-12.



LOB

Large Object (LOB) data types refer to large data items such as text, graphic images, video clips, and sound waveforms. LOB data types allow efficient, random, piecewise access to this data.





Variables

A variable is nothing but a name given to a storage area that our programs can manipulate.

Rules are:

1. The name of a PL/SQL variable consists of a letter
2. optionally followed by more letters, numerals, dollar signs, underscores, and number signs
3. should not exceed 30 characters.
4. By default, variable names are not case-sensitive.
5. You cannot use a reserved PL/SQL keyword as a variable name.

Variable declaration:

When you declare a variable, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.

Variable is declared in declaration section.

SYNTAX:

variable\_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial\_value]

* variable\_name is a valid identifier in PL/SQL
* datatype must be a valid PL/SQL data type or any user defined data type
* [NOT NULL] is constraint
* [:= | DEFAULT initial\_value] is technique to initialize a value.

Eg:

sales number(10, 2);

pi CONSTANT double precision := 3.1415;

name varchar2(25);

address varchar2(100);

When you provide a size, scale or precision limit with the data type, it is called a constrained declaration.

Constrained declarations require less memory than unconstrained declarations. For example −

sales number(10, 2);

name varchar2(25);

address varchar2(100);

Variable initialization

Whenever you declare a variable, PL/SQL assigns it a default value of NULL.

If you want to initialize a variable with a value other than the NULL value, you can do so during the declaration, using either of the following −

* The **DEFAULT** keyword
* The **assignment** operator

For example −

counter binary\_integer := 0;

greetings varchar2(20) DEFAULT 'Have a Good Day';

You can also specify that a variable should not have a **NULL** value using the **NOT NULL** constraint.

If you use the NOT NULL constraint, you must explicitly assign an initial value for that variable.

It is a good programming practice to initialize variables properly otherwise, sometimes programs would produce unexpected results.

Variable Scope

PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block.

If a variable is declared within an inner block, it is not accessible to the outer block.

However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks.

There are two types of variable scope −

* Local variables − Variables declared in an inner block and not accessible to outer blocks.
* Global variables − Variables declared in the outermost block or a package.

Following example shows the usage of Local and Global variables in its simple form −

DECLARE

-- Global variables

num1 number := 95;

num2 number := 85;

BEGIN

dbms\_output.put\_line('Outer Variable num1: ' || num1);

dbms\_output.put\_line('Outer Variable num2: ' || num2);

DECLARE

-- Local variables

num1 number := 195;

num2 number := 185;

BEGIN

dbms\_output.put\_line('Inner Variable num1: ' || num1);

dbms\_output.put\_line('Inner Variable num2: ' || num2);

END;

END;

/

When the above code is executed, it produces the following result −

Outer Variable num1: 95

Outer Variable num2: 85

Inner Variable num1: 195

Inner Variable num2: 185

PL/SQL procedure successfully completed.

Assigning SQL Query Results to PL/SQL Variables

You can use the SELECT INTO statement of SQL to assign values to PL/SQL variables. For each item in the SELECT list, there must be a corresponding, type-compatible variable in the INTO list. The following example illustrates the concept. Let us create a table named CUSTOMERS –

DECLARE

c\_id customers.id%type := 1;

c\_name customers.name%type;

c\_addr customers.address%type;

c\_sal customers.salary%type;

BEGIN

SELECT name, address, salary INTO c\_name, c\_addr, c\_sal

FROM customers

WHERE id = c\_id;

dbms\_output.put\_line

('Customer ' ||c\_name || ' from ' || c\_addr || ' earns ' || c\_sal);

END;

/

When the above code is executed, it produces the following result −

Customer Ramesh from Ahmedabad earns 2000

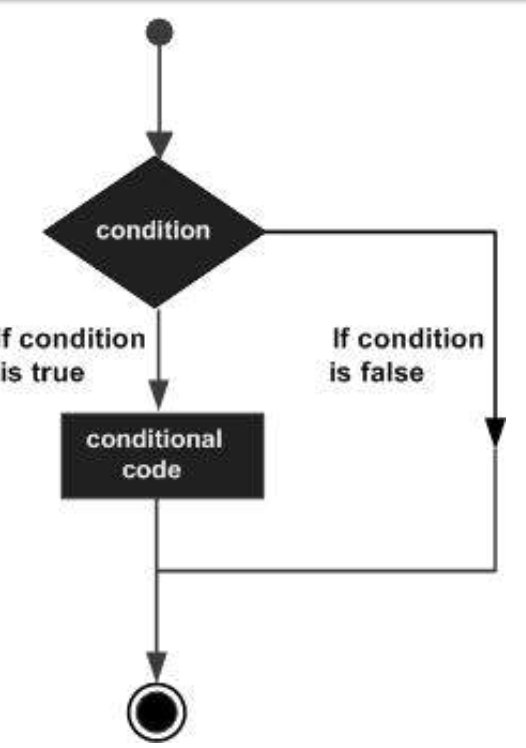
Constants and Literals

A constant holds a value that once declared, does not change in the program. A constant declaration specifies its name, data type, and value, and allocates storage for it. The declaration can also impose the NOT NULL constraint.

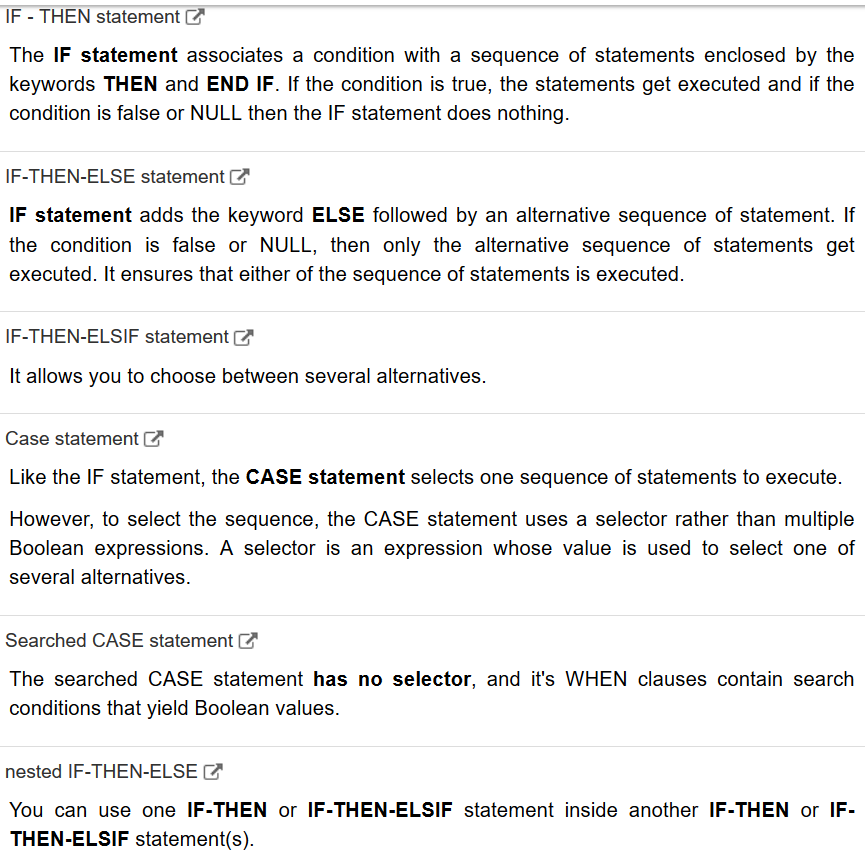
<https://www.tutorialspoint.com/plsql/plsql_constants.htm>

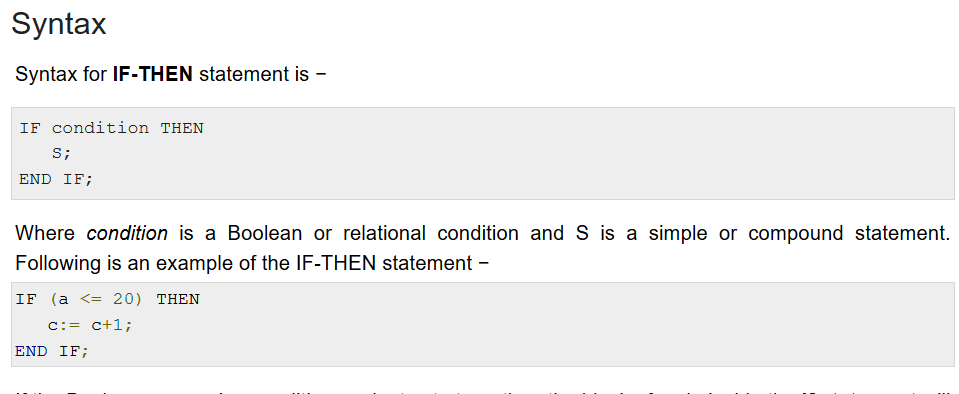
Operators

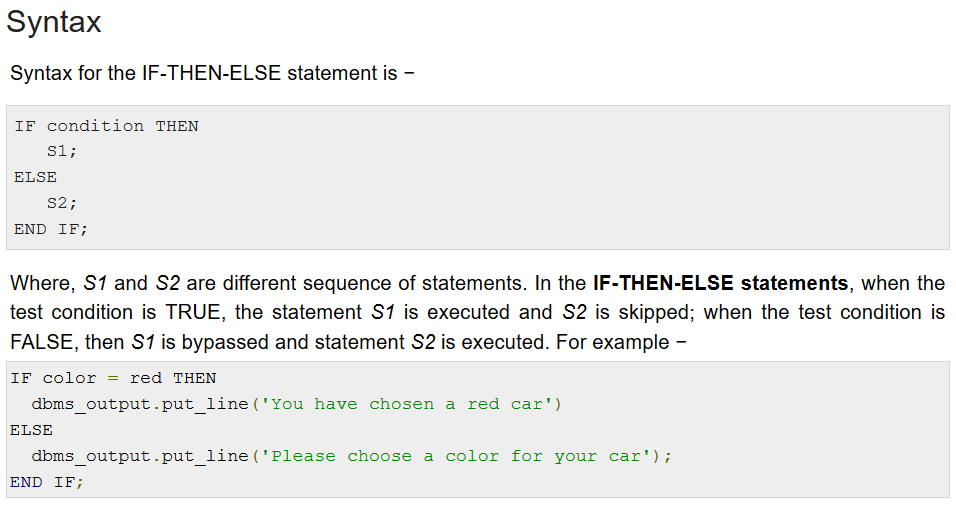
Conditions

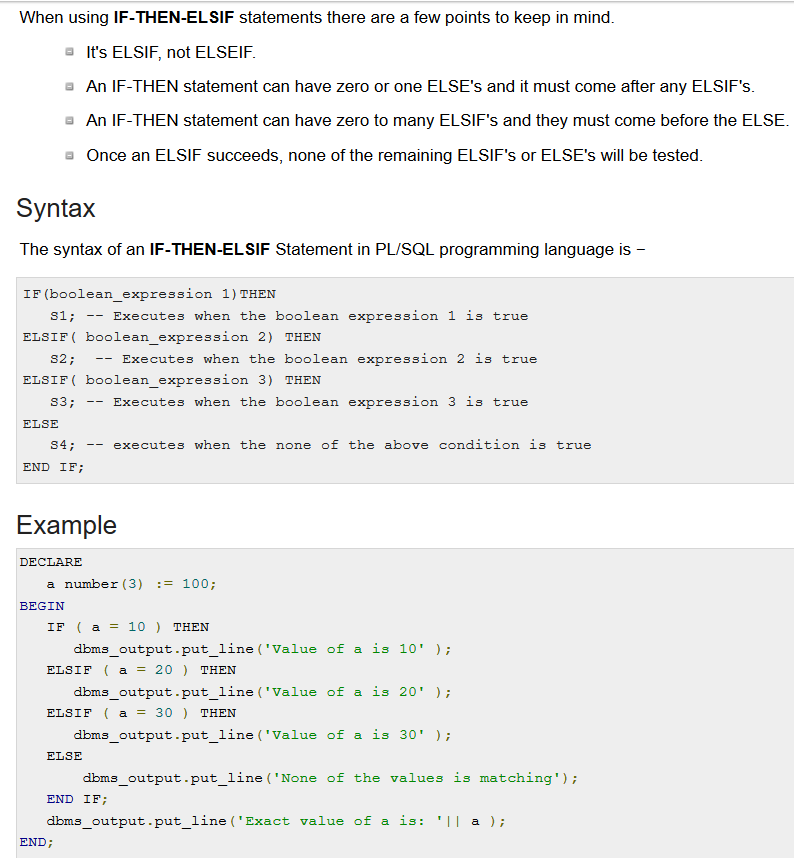


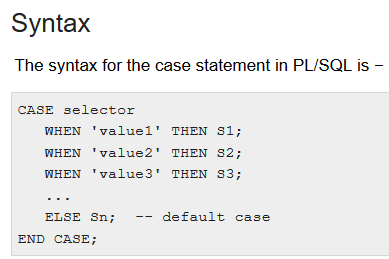
PL/SQL programming language provides following types of decision-making statements.

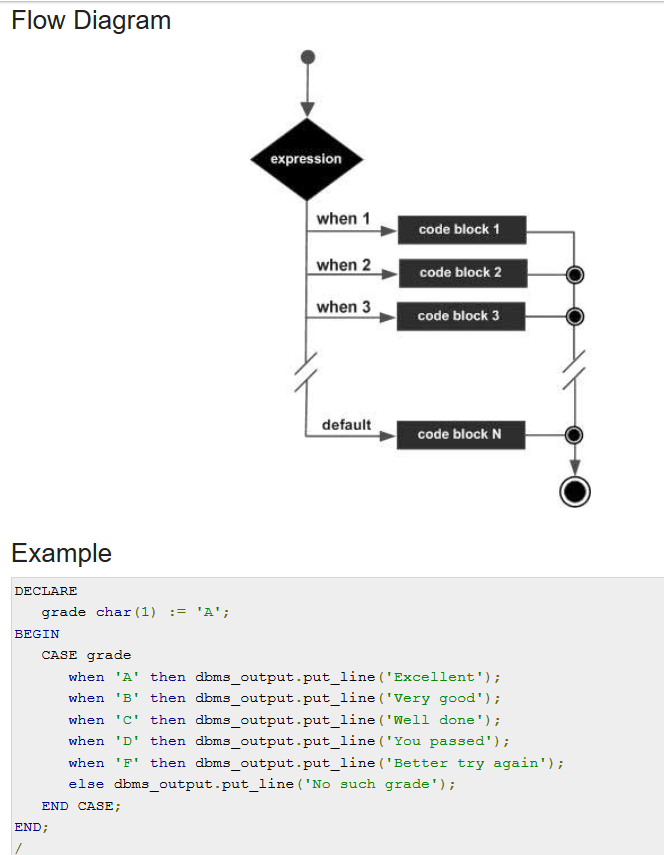


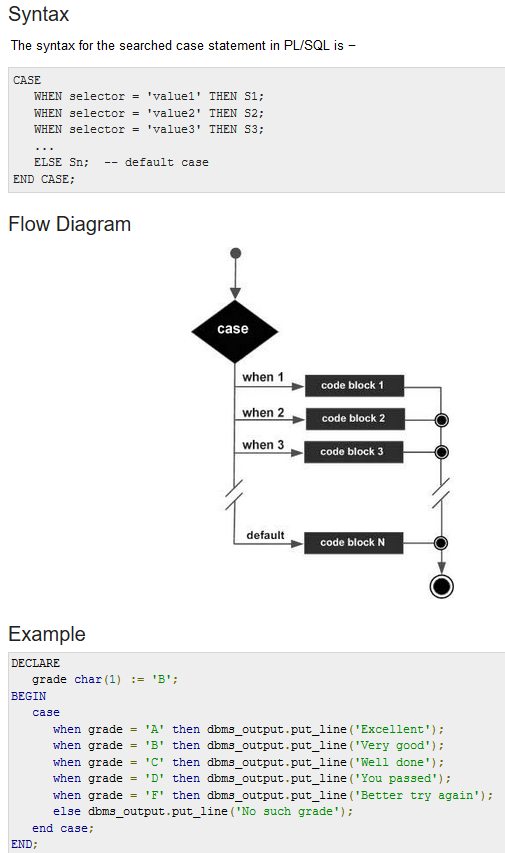


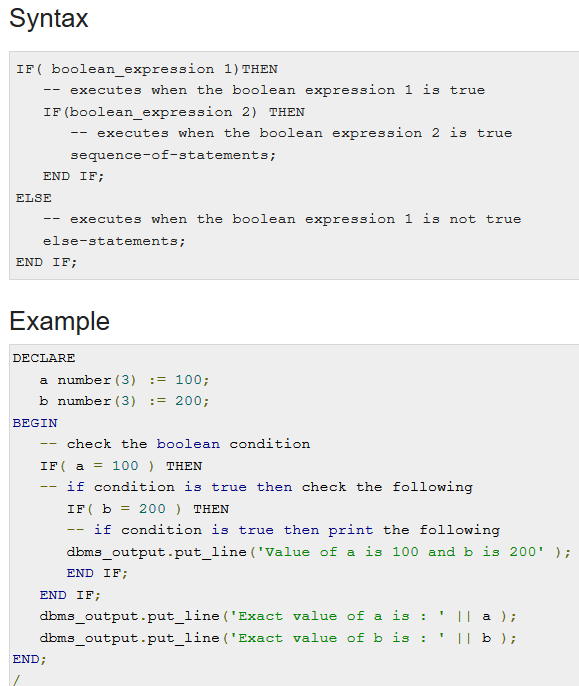




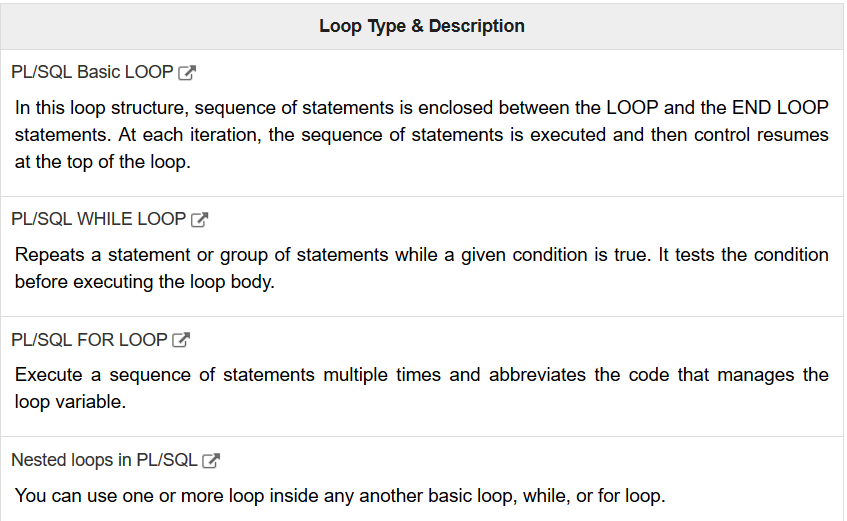


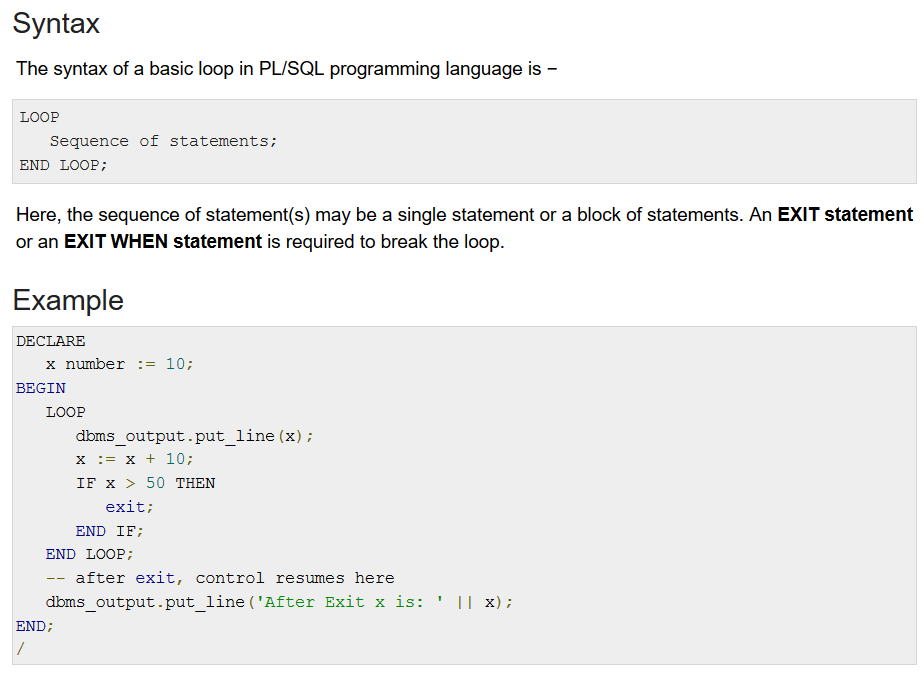


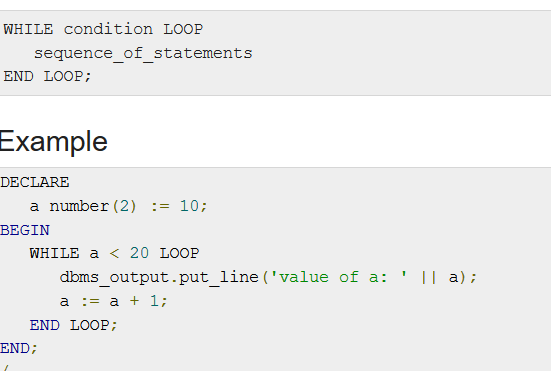


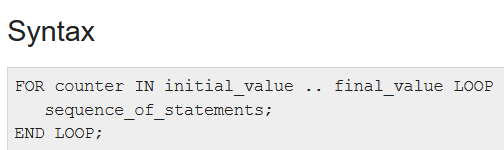


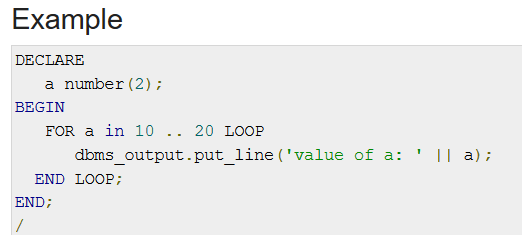
Loops

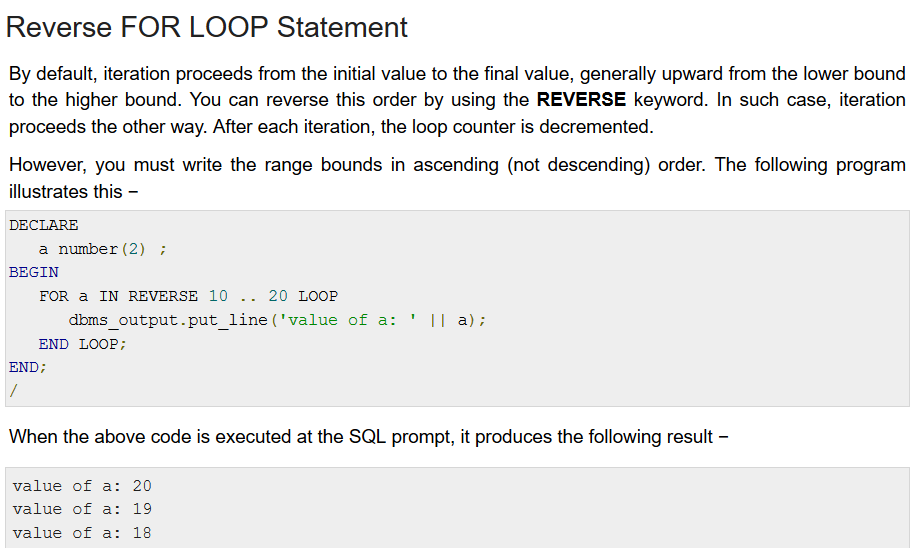


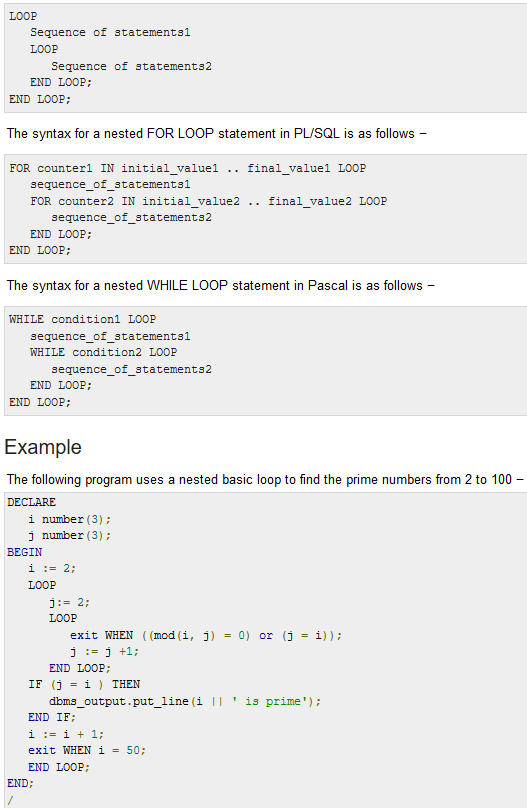


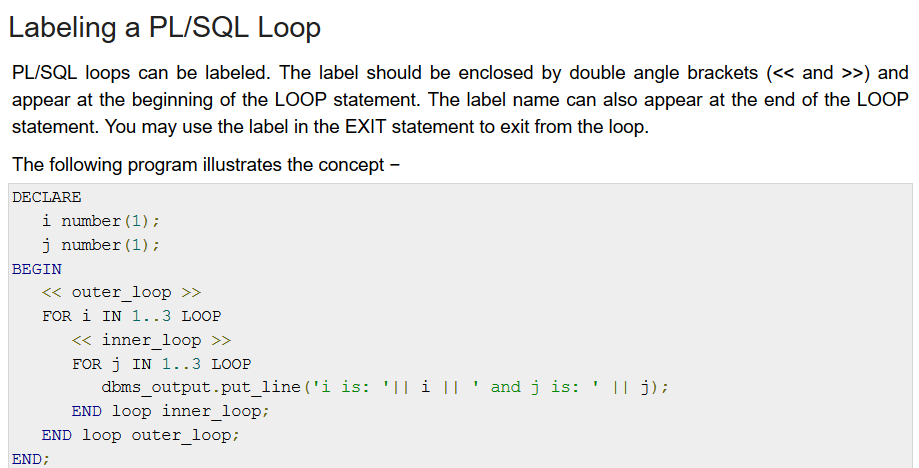












Strings

Arrays

Procedures

Functions

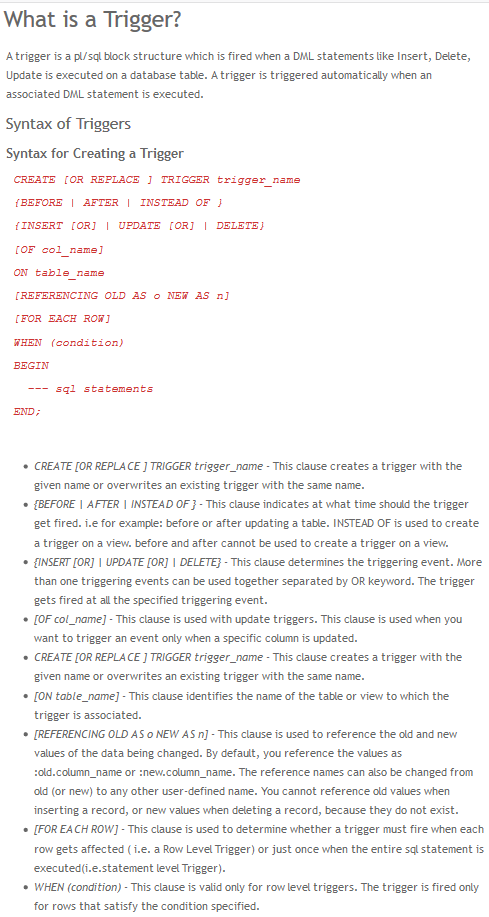
Cursors

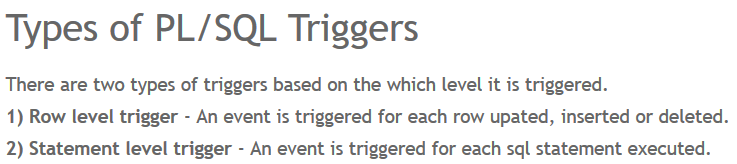
Records

Exceptions

Triggers

<https://plsql-tutorial.com/plsql-triggers.htm>





Eg:

Packages

Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms. A package will have two mandatory parts −

* Package specification
* Package body or definition

Package specification: The specification is the interface to the package. It just **DECLARES** the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. All objects placed in the specification are called **public** objects. Any subprogram not in the package specification but coded in the package body is called a **private** object.

Eg:

CREATE PACKAGE cust\_sal AS

PROCEDURE find\_sal(c\_id customers.id%type);

END cust\_sal;

Package body: The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package. The **CREATE PACKAGE BODY** Statement is used for creating the package body.

Eg:

CREATE OR REPLACE PACKAGE BODY cust\_sal AS

PROCEDURE find\_sal(c\_id customers.id%TYPE) IS

c\_sal customers.salary%TYPE;

BEGIN

SELECT salary INTO c\_sal

FROM customers

WHERE id = c\_id;

dbms\_output.put\_line('Salary: '|| c\_sal);

END find\_sal;

END cust\_sal;

Collections

Transactions

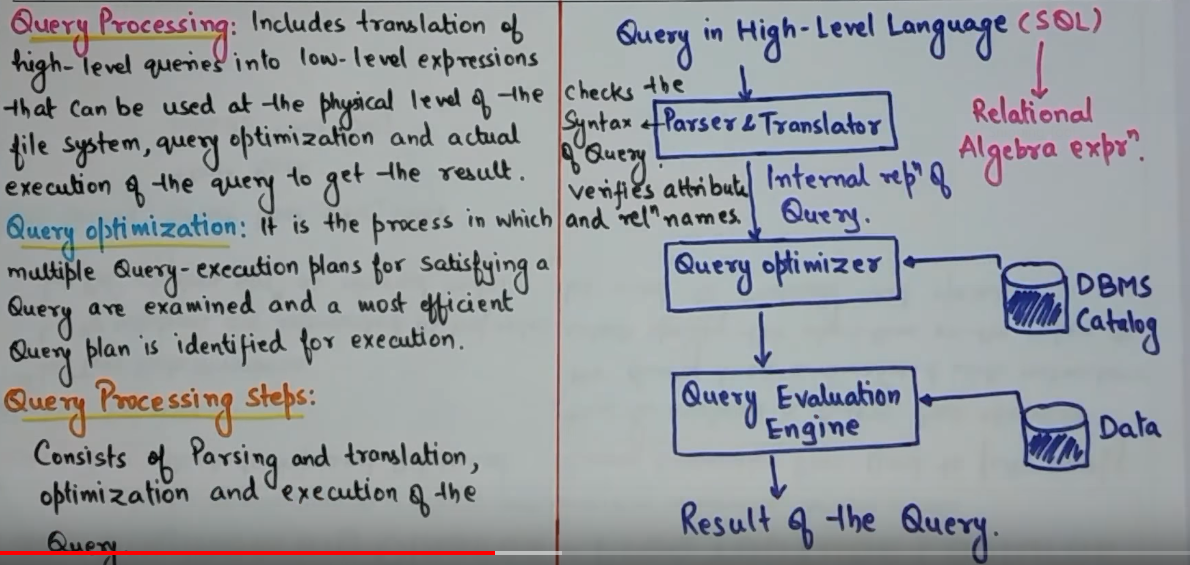
Date & Time

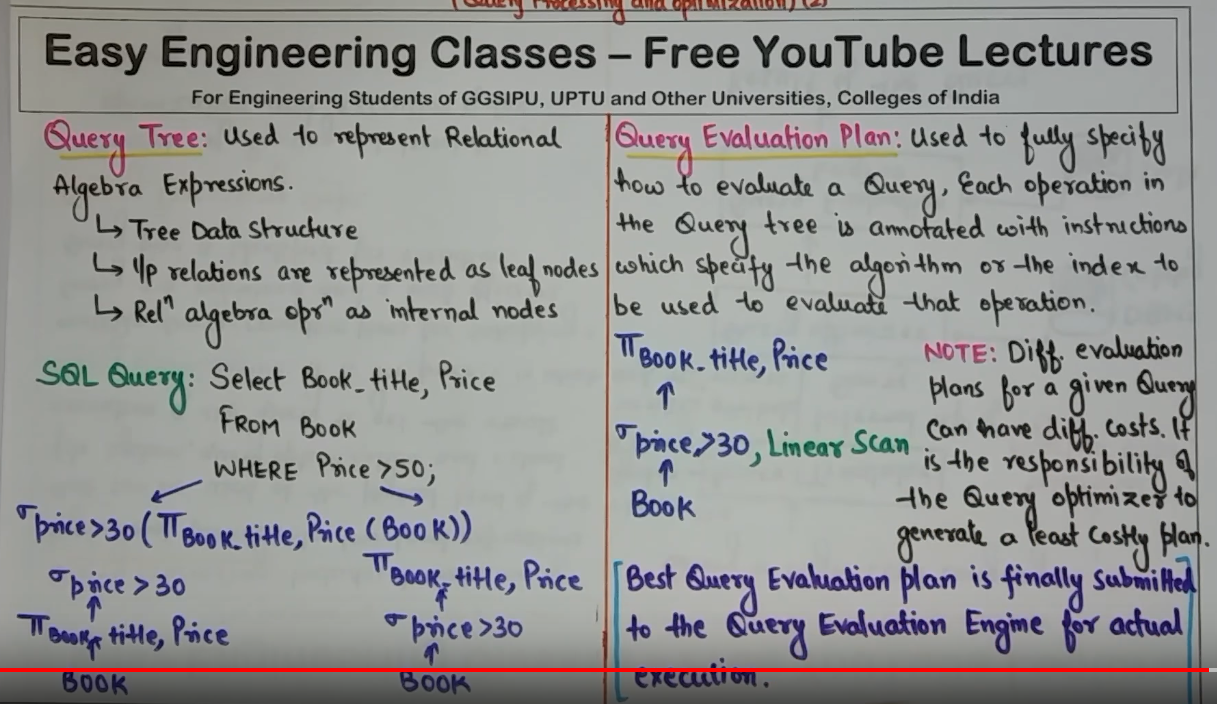
DBMS Output

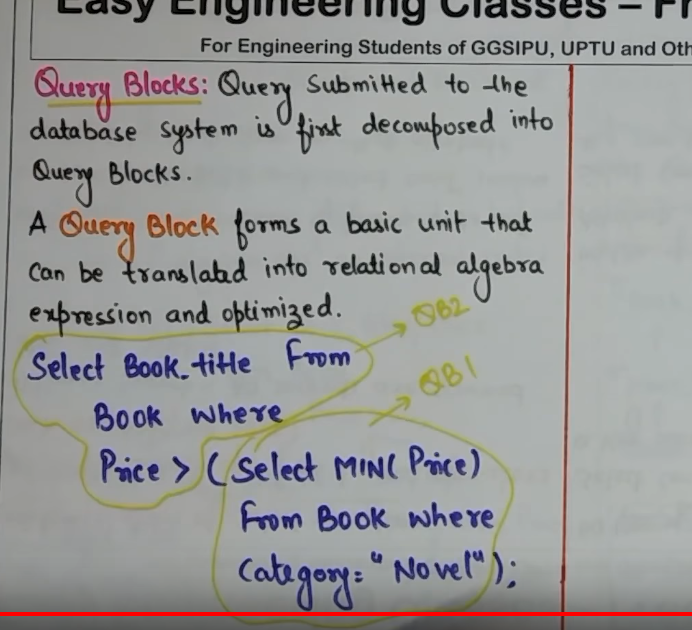
Chapter 2

<https://www.javatpoint.com/dbms-transaction-processing-concept>

Chapter 3







<https://www.geeksforgeeks.org/sql-query-processing/>

<https://www.slideshare.net/ravi_LCET/query-processing-and-optimization-updated>

<https://www.slideshare.net/ravi_LCET/cost-estimation-for-query-optimization>