**INSERT INTO** table (a,b,c) VALUES (‘a’,’b’,’c’) - special insert: NULL, SYSDATE, USER, CURRENT\_DATE, use

TO\_DATE to specify different format(reg. DD-MMM-YYYY), copy values from another table do not use VALUES

**UPDATE** table SET VALUES [WHERE conditions], are able to use SELECT in both SET and WHERE

**DELETE** table[WHERE conditions], no specify WHERE = delete whole table, remove table structure

**TRUNCATE** **TABLE** table, delete all values and keep table structure **DML**: no commit **DDL/DCL**: auto commit

**COMMIT**: save data **SAVEPOINT**: create a restore point **ROLLBACK / ROLLBACK TO SAVEPOINT**: undo

**CREATE TABLE** table(column\_name / types(varchar2, char[size], number[p,s], date, long] / optional(primary keys, not

null, unique, foreign key, check), you can add **CONSTRAINTS** below the column name or end of table creation. You can either copy one table to another, copy a part of a table to another, just make sure the new table is already created or the copy as subquery (CREATE TABLE table AS SELECT (columns) FROM table [WHERE conditions]), must give ALIAS to SELECT where included expression such as multiplication.

**ALTER TABLE** table ADD/DROP/MODIFY columns\_name types;

**PL/SQL** format:

* **Block Types** (add to the top): **Anonymous**: nothing, **Procedure**: PROCEDURE [name] IS , **Function**: FUNCTION [name] RETURN [datatype] IS.
* **DECLARE** [optional] (create var).
* Syntax: var\_name / data types [NOT NULL] [:= / DEFAULT values ], long values use q’!values!’ or q’[values]’. Avoid create the var\_name similar or identical to the existing one.
* **SCALAR** is a normal variable, can hold a single value, exp: DECLARE v\_emp\_job VARCHAR2(9) **REFERENCE**: hold a pointer to a storage location (usually known as %TYPE, which copy the data type), exp: DECLARE emp\_lname employees.last\_name%TYPE **COMPOSITE**: contain both **SCALAR** and **REFERENCE**, exp: DECLARE emp\_lname employees.last\_name%TYPE := ‘John’.
* Prompt the user to enter the value to var, exp: v\_empno NUMBER(6) := &empno so each time, it will prompt the user to enter the value, && to remember the value inserted.
* Quote variable (var include space) must be in lowercase
* **[BIND VARIABLE]**: Does not create in Declare, simply as VARIABLE b\_result NUMBER. This var can be reuse outside of PL/SQL scoop
* **BEGIN** [mandatory] (must have at least one SQL or PL/SQL statement).
* If you want to display the information has been change, add **DBMS\_OUTPUT.PUT\_LINE** (columns\_name\_changed).
* All SQL function except **DECODE** and Group function(**AVG, MIN, MAX, COUNT, SUM, STDDEV, VARIANCE**) are available in PL/SQL.
* Implicit is applied when the operation automatically convert data type to other data type, usually VARCHAR to specific data type such as Number or Date. Explicit require another function (**TO\_CHAR, TO\_DATE, TO\_NUMBER, TO\_TIMESTAMP**), exp: date DATE := TO\_DATE(‘February 02, 2000’, ‘Month DD, YYYY’).
* Operator order: Exponential (\*\*), Arithmetic ( + , - , \* , / ), Concatenation (+ , - , | | ), Parentheses (= , < , > , <= , >= , <> , != , -= , ^= , IS NULL, LIKE, BETWEEN, IN), Logical (NOT, AND, OR).
* To copy existing data into newly created var, simply us **SELECT** column(s) **INTO** var. This SELECT INTO can use Group Function.
* To see number of row is being effect, use SQL%ROWCOUNT, if the most recent statement returned at least one row, SQL%FOUND == true, otherwise, SQL%NOTFOUND == true
* **EXCEPTION** [optional] (where to be specific)
* **END**; [mandatory]. , add **/** if running another statement
* **[OPTIONAL]**: **SET AUTO PRINT ON**: Auto display the bind variables values, **SET VERIFY ON/OFF**

**IF Statement:** **IF** condition **THEN** statements; [**ESLIF** condition **THEN** statement] / [**ELSE** statements;] **END IF**:

**CASE Expression**: **CASE** column **WHEN** expression **THEN** result [ELSE result] **END**;

**LOOP Statement**: **LOOP** … **EXIT** [WHEN condition] **END LOOP** , WHILE condition **LOOP** … **END LOOP** , **FOR** counter IN [REVERSE] lower\_bound..upper\_bound **LOOP** … **END LOOP.** Loop can be nested

Copy the entire attributes + data type from a table: var test reference%**ROWTYPE,** like create a record for less work

Create a record: **TYPE** name **IS RECORD** (identifier type\_name, [dentifier type\_name,...])

Instead of getting all values + attribute like record, you can use **CURSOR** to get part of it. Syntax: **CURSOR** cursor\_name **IS SELECT FROM WHERE; %ISOPEN** == true if is open,%NOTFOUND == true if does not return a row, %FOUND == true if return a row, %ROWCOUNT return number of rows affected



  
