Views

CREATE [OR REPLACE] VIEW <vName> AS <Query>;

Triggers

Exec Order:

1. BEFORE (Statement-Level);
2. ∀ affected records: (a) BEFORE (Row-Level); (b) Event (Row; (c) AFTER (Row);
3. AFTER (Statement);

Variables

- Declaration like normal but with a semicolon: <varName> <varType>;
- Can also declare Cursors: CURSOR <cName> IS <query>;
    - Parameterized Cursor: CURSOR <cName> (<param> <type>, …) IS…;
- Set variables from a table by SELECT <column[s]> INTO <varName[s]>...;
    - System Variables: SELECT sysdate INTO temp FROM Dual;

Code Body

- If statements are explicit: IF (<condition>) THEN <code> END IF;
- Looping through a cursor: FOR row IN cName LOOP <code> END LOOP;
- In Row-Level, may get :new and :old variables (depending on triggering operation)
- Output to console: dbms_output.put_line('message');
- Raise Error: RAISE_APPLICATION_ERROR(-20001, 'errMessage');

```
CREATE [OR REPLACE] TRIGGER <TriggerName>
[BEFORE | AFTER] [INSERT | UPDATE [OF <columnName>]] ON <TableName>
[FOR EACH ROW] -- "FOR EACH STATEMENT" implicit if omitted
[DECLARE
        <Decleration>; ...]
BEGIN
        <PL/SQL Code>
END; /
```

Procedures & Functions

- Procedures can't output except via output parameters
- Invoke procedures: EXEC <pName>(<params>); (May need to SET serveroutput on;)
- Invoke Functions anywhere with <fName>(<params>), including in WHERE clause
- Access variables declared with the name of procedure or function: <name>.<varName>

```
CREATE [OR REPLACE] [PROCEDURE | FUNCTION] <name>
        [(<paramName> [IN | OUT] <paramType>, …)]
        [RETURN <returnType] -- only if this is a function
        IS
        [<varDeclaration>;...]
BEGIN
        <PL/SQL Code>
END <pName>; /
```