1==========BLOCKS ========

```
DO $$
<<first_block>>
DECLARE
  counter integer := 0;
BEGIN
   counter := counter + 1;
   RAISE NOTICE 'The current value of counter is %', counter;
END first_block $$;
```

2==========

```
DO $$
<<outer_block>>
DECLARE
  counter integer := 0;
BEGIN
   counter := counter + 1;
   RAISE NOTICE 'The current value of counter is %', counter;

   DECLARE
      counter integer := 0;
   BEGIN
      counter := counter + 10;
      RAISE NOTICE 'The current value of counter in the subblock is %', counter;
      RAISE NOTICE 'The current value of counter in the outer block is %', outer_block.counter;
   END;

   RAISE NOTICE 'The current value of counter in the outer block is %', counter;

END outer_block $$;
```

3==========VARIABLES =========

```
DO $$
DECLARE
   counter integer := 1;
   first_name varchar(50) := 'John';
   last_name varchar(50) := 'Doe';
   payment numeric(11,2) := 20.5;
BEGIN
   RAISE NOTICE '% % % has been paid % USD', counter, first_name, last_name, payment;
END $$;
```

4========== CONSTANTS ==========

```
DO $$
DECLARE
   VAT CONSTANT numeric := 0.1;
   net_price numeric := 20.5;
BEGIN
   RAISE NOTICE 'The selling price is %', net_price * ( 1 + VAT );
END $$;
```

5========== MESSAGES ==========

```
DO $$
BEGIN
  RAISE INFO 'information message %', now() ;
  RAISE LOG 'log message %', now();
  RAISE DEBUG 'debug message %', now();
  RAISE WARNING 'warning message %', now();
  RAISE NOTICE 'notice message %', now();
END $$;
```

6==========
```
DO $$
DECLARE
 email varchar(255) := 'info@ced.tuc.gr';
BEGIN
  -- check email for duplicate
  -- ...
  -- report duplicate email
  RAISE EXCEPTION 'Duplicate email: %', email
 USING HINT = 'Check the email again';

-- RAISE vivision_by_zero
-- RAISE SQLSTATE ‘22012’
END $$;
```

7==========FUNCTIONS

```
CREATE OR REPLACE FUNCTION get_sum(
 a NUMERIC,
 b NUMERIC)
RETURNS NUMERIC AS $$
BEGIN
 RETURN a + b;
END; $$

LANGUAGE plpgsql;
```

8==========

```
CREATE OR REPLACE FUNCTION hi_lo(
 a NUMERIC,
 b NUMERIC,
 c NUMERIC,
 OUT hi NUMERIC,
 OUT lo NUMERIC)
AS $$
BEGIN
 hi := GREATEST(a,b,c);
 lo := LEAST(a,b,c);
END; $$
LANGUAGE plpgsql;
```

9==========

```
CREATE OR REPLACE FUNCTION square(
 INOUT a NUMERIC)
AS $$
BEGIN
 a := a * a;
END; $$
LANGUAGE plpgsql;
```

10==========

```
CREATE OR REPLACE FUNCTION sum_avg(
 VARIADIC list NUMERIC[],
 OUT total NUMERIC,
 OUT average NUMERIC)
AS $$
DECLARE
        x integer;
BEGIN

  total := 0;
  FOREACH x IN ARRAY list LOOP
                total := total + x;
  END LOOP;

  average := total / array_length(list, 1);
END; $$
LANGUAGE plpgsql;
```

11==========

```
CREATE OR REPLACE FUNCTION get_student (p_pattern VARCHAR)
 RETURNS TABLE (
 student VARCHAR,
 am_number NUMERIC
)
AS $$
BEGIN
 RETURN QUERY SELECT
 (name || surname)::varchar,
 cast( am as NUMERIC)
 FROM
 "Student"
 WHERE
 surname LIKE p_pattern ;
END; $$

LANGUAGE 'plpgsql';
```

12==========

```
CREATE OR REPLACE FUNCTION get_student (p_pattern VARCHAR,p_year INTEGER)
 RETURNS TABLE (
 student_name VARCHAR,
 first_year INTEGER
) AS $$
DECLARE
    var_r record;
BEGIN
 FOR var_r IN(SELECT surname,  entry_date FROM "Student"  WHERE surname LIKE
p_pattern AND entry_date > (p_year || '-01-01')::DATE)
 LOOP
        student_name := lower(var_r.surname) ;
           first_year := date_part('year', var_r.entry_date);
        RETURN NEXT;
 END LOOP;
END; $$
LANGUAGE 'plpgsql';
```

13==========CURSORS

```
CREATE OR REPLACE FUNCTION get_course_titles(p_year INTEGER)
   RETURNS text AS $$
DECLARE
 titles TEXT DEFAULT '';
 rec_course RECORD;
 cur_courses CURSOR(p_year INTEGER)
 FOR SELECT * FROM "Course" WHERE  typical_year = p_year;
BEGIN
   -- Open the cursor
   OPEN cur_courses(p_year);

   LOOP
    -- fetch row into the film
      FETCH cur_courses INTO rec_course;
    -- exit when no more row to fetch
      EXIT WHEN NOT FOUND;

    -- build the output
      IF rec_course.course_code LIKE 'ÐËÇ%' THEN
         titles := titles || ',' || rec_course.course_title || ':' || rec_course.typical_year;
      END IF;
```

```
    END LOOP;

    -- Close the cursor
    CLOSE cur_courses;

    RETURN titles;
END; $$

LANGUAGE plpgsql;
```