اساسيات كتابة برنامج pl/sql

1️⃣ بيئة التنفيذ (Environment)

قبل أي شيء، لازم البرنامج يتنفذ داخل بيئة تدعم **PL/SQL** زي:

- **SQL Developer**
- **SQL*Plus**
- **Oracle Forms** أو **JDeveloper**

📌 في SQL Developer، لازم تفعلي إظهار المخرجات:

```sql
SET SERVEROUTPUT ON;
```

- دي بتسمح بعرض النصوص من خلال `DBMS_OUTPUT.PUT_LINE`.

2️⃣ تركيب الوحدة البرمجية (PL/SQL Block Structure)

أي برنامج PL/SQL يتكوّن من 3 أقسام رئيسية:

القسم	الكلمة المحجوزة	الغرض
قسم التعريف	`DECLARE`	تعريف المتغيرات، المؤشرات، الثوابت، إلخ.
قسم التنفيذ	`BEGIN`	كتابة الأوامر القابلة للتنفيذ (SQL أو PL/SQL).
قسم الأخطاء	`EXCEPTION`	التعامل مع الأخطاء إن وجدت.

💡 بعض الأقسام اختيارية:

- `DECLARE` → اختياري (لو ما عندك متغيرات تعرفيها)
- `EXCEPTION` → اختياري
- `BEGIN` و `END;` → إجباري

3 أنواع الوحدات البرمجية

- **كتلة غير مسماة (Anonymous Block)** → تنفذ فورًا، لا تُخزن.
- **كتلة مسماة (Named Block)** → زي الإجراء (`Procedure`) أو الدالة (`Function`)، تُخزن في قاعدة البيانات.

---

4 المتغيرات (Variables)

- لازم تُعرف داخل `DECLARE` .
- صيغة التعريف:

```plsql
variable_name datatype [:= initial_value];
```

- النصوص داخل `' '` مش `" "` .
- التواريخ لازم تتحول بـ `TO_DATE` .

مثال:

V_Name VARCHAR2(20) := 'Ali';

V_Birth DATE := TO_DATE('04-Apr-1965', 'DD-Mon-YYYY');

5 تسمية البلوكات (Labeling Blocks)

لما يكون عندك بلوك داخل بلوك، وعايزة توصلي لمتغير من البلوك الخارجي، لازم تسمي البلوك الخارجي:

```pl
<<outer>>
DECLARE
   x NUMBER := 3;
BEGIN
   DECLARE
      x NUMBER := 5;
   BEGIN
      DBMS_OUTPUT.PUT_LINE(x); -- 5
      DBMS_OUTPUT.PUT_LINE(outer.x); -- 3
   END;
END;
```

💡 الفايدة: تقدر تفرق بين المتغير الداخلي والخارجي.

6 التعامل مع التواريخ

- ما تكتبي التاريخ كنص عادي، استخدمي:

```plsql
TO_DATE('04-Apr-1965', 'DD-Mon-YYYY')
```

- لو عايزة تعرضه كنص:

```pl
TO_CHAR(my_date, 'DD-Mon-YYYY')
```

---

7 الحزم الجاهزة (Packages) — مثل `DBMS_OUTPUT`

- `DBMS_OUTPUT.PUT_LINE` → لطباعة النصوص أو القيم.
- ما تشتغل إلا إذا فعلتي:

```sql
SET SERVEROUTPUT ON;
```

---

8 الفواصل المنقوطة `;`

- نهاية كل أمر لازم يكون فيه `;`
- لكن `DECLARE` و `BEGIN` و `EXCEPTION` ما بيجوا معاهم فاصلة.

9 تنفيذ الكود

- بعد آخر `END;` في الكتلة، لو بتنفذي في SQL*Plus أو نافذة الأوامر في SQL Developer، لازم تضيفي `/` في سطر جديد:

```sql
END;
/
```

عشان يقول للبيئة: "نفذ البلوك كامل".

🔟 الأخطاء الشائعة وأسبابها

الخطأ	السبب	الحل
PLS-00103 عند النصوص	استخدام " " بدل ' '	استخدم `'Ali'` للنصوص
PLS-00103 عند التاريخ	إدخال تاريخ كنص بدون TO_DATE	استخدم `TO_DATE('04-Apr-1965','DD-Mon-YYYY')`
متغير غير معروف	تعريفه في بلوك آخر بدون Label	استخدم اسم البلوك أو عرّفه في نفس البلوك
لا يظهر الإخراج	ما فعلت `SET SERVEROUTPUT ON`	فعل الأمر قبل التنفيذ

مثال لتعريف المتغيرات بأنواع مختلفة فيPL/SQL.

```
DECLARE
  -- متغير نصي (String)
  v_name VARCHAR2(50) := 'Ali';
  -- متغير عددي صحيح
  v_age NUMBER := 30;
  -- متغير عشري (Decimal)
  v_salary NUMBER(8,2) := 4500.75;
  -- متغير تاريخ
  v_join_date DATE := TO_DATE('2023-08-10', 'YYYY-MM-DD');
  -- متغير منطقي (Boolean)
  v_is_active BOOLEAN := TRUE;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
  DBMS_OUTPUT.PUT_LINE('Age: ' || v_age);
```

```
  DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
  DBMS_OUTPUT.PUT_LINE('Join Date: ' || TO_CHAR(v_join_date, 'DD-MM-YYYY'));
  DBMS_OUTPUT.PUT_LINE('Is Active: ' || CASE WHEN v_is_active THEN 'Yes' ELSE 'No' END);
END;
/
```

2.مثال على بلوك داخلي (Nested Block) وبلوكة خارجي (Outer Block)

```
DECLARE
  v_outer_var NUMBER := 10; -- متغير في البلوك الخارجي
BEGIN
  DBMS_OUTPUT.PUT_LINE('Outer block variable: ' || v_outer_var);
  -- بلوك داخلي
  DECLARE
    v_inner_var NUMBER := 20; -- متغير خاص بالبلوك الداخلي
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Inner block variable: ' || v_inner_var);
    DBMS_OUTPUT.PUT_LINE('Accessing outer block variable from inner block: ' || v_outer_var);
  END;
```

```
-- لأنه خاص بالبلوك الداخلي (v_inner_var) لا يمكن هنا الوصول للمتغير الداخلي
 -- DBMS_OUTPUT.PUT_LINE('Trying to print inner var: ' ||
v_inner_var); -- هذا خطأ
END;
/
```

مثال على الجمل الشرطية(IF) (CASE).

```
DECLARE
 v_score NUMBER := 75;
 v_grade CHAR(1);
BEGIN
 -- جملة شرطية if
 IF v_score >= 90 THEN
  v_grade := 'A';
 ELSIF v_score >= 80 THEN
  v_grade := 'B';
 ELSIF v_score >= 70 THEN
  v_grade := 'C';
 ELSE
  v_grade := 'F';
 END IF;
 DBMS_OUTPUT.PUT_LINE('Grade using IF: ' || v_grade);
 -- جملة شرطية CASE
 CASE
```

```
    WHEN v_score >= 90 THEN v_grade := 'A';
    WHEN v_score >= 80 THEN v_grade := 'B';
    WHEN v_score >= 70 THEN v_grade := 'C';
    ELSE v_grade := 'F';
  END CASE;
  DBMS_OUTPUT.PUT_LINE('Grade using CASE: ' || v_grade);
END;
/
```