

ORACLE PL/SQL · PERTEMUAN 10 / 14

Control Structure, Loop & Exception

Memilih cabang, membatasi pengulangan dan menangani kegagalan dengan sengaja.

Instalasi & SQL → Mini Project → PL/SQL → Final Project · Oracle XE lokal atau FreeSQL dengan login

Versi PDF materi pertemuan 10. Contoh dan tabel dapat dibaca di PDF. Kuis dapat dijalankan di halaman web: [Buka materi daring](#).

Tujuan pembelajaran

1. Menulis IF, ELSIF, nested IF serta CASE dengan cabang lengkap.
2. Memilih LOOP, WHILE dan FOR serta menempatkan EXIT secara aman.
3. Menangani NO_DATA_FOUND, TOO_MANY_ROWS dan ZERO_DIVIDE secara spesifik.
4. Mendefinisikan exception sendiri dan memakai RAISE_APPLICATION_ERROR untuk validasi.
5. Menjelaskan alur sesudah exception tanpa menelan error atau mengambil alih transaksi pemanggil.

Pokok bahasan / praktikum

- IF THEN ELSE
- ELSIF
- Nested IF
- Simple CASE
- Searched CASE
- LOOP
- WHILE LOOP
- FOR LOOP
- EXIT dan EXIT WHEN
- NO_DATA_FOUND
- TOO_MANY_ROWS
- ZERO_DIVIDE

- Custom exception dan RAISE
- RAISE_APPLICATION_ERROR
- Validasi nilai uang

Prasyarat: Selesaikan modul 02/03 atau jalankan setup bersama sekali pada schema latihan sendiri. Gunakan Oracle XE lokal melalui SQL Developer ATAU Oracle FreeSQL setelah login pada schema pribadi yang diizinkan; tidak harus memakai keduanya. Jangan menjalankan ulang CREATE TABLE/seed pada objek yang sudah ada. Selesaikan modul SQL sebelumnya; modul 10 juga memerlukan dasar blok modul 09.

Pilih salah satu lingkungan praktikum

Oracle XE: XE lokal dengan SQL Developer: cocok untuk instalasi, koneksi, user/schema, privilege, tablespace serta latihan SQL/PLSQL.

FreeSQL: FreeSQL dengan login: masuk menggunakan akun sendiri dan gunakan schema latihan yang diizinkan untuk SQL/PLSQL. Tidak perlu memasang XE untuk menyelesaikan latihan non-administratif.

Pengumpulan tugas menerima hasil dari salah satu lingkungan. Pastikan objek prasyarat tersedia dan tetap gunakan schema latihan, bukan schema produksi.

AMATI → PREDIKSI → JELASKAN

Mulai dari pertanyaan

1. Apa yang terjadi ketika kondisi IF bernilai NULL?
2. Bagaimana memastikan loop tidak berjalan tanpa akhir?
3. Kapan kesalahan harus ditangani lokal dan kapan perlu dikirim kepada pemanggil?

Uji pemahaman awal

1. WHILE berbeda dari basic LOOP karena ...

- WHILE selalu berjalan sekali
- WHILE memeriksa kondisi sebelum badan
- LOOP tidak boleh EXIT
- WHILE hanya untuk SELECT

2. Exception bawaan untuk pembagian dengan nol adalah ...

- NO_DATA_FOUND
- TOO_MANY_ROWS
- ZERO_DIVIDE
- CASE_NOT_FOUND

3. Nomor valid untuk RAISE_APPLICATION_ERROR adalah ...

- -20001
- 20001
- -100
- -30001

PAHAMI KONSEP · BACA TERSTRUKTUR · TERAPKAN

Teori dan pembahasan

Gunakan percabangan untuk keputusan, loop untuk pekerjaan berulang, dan exception untuk alur kegagalan. Setiap mekanisme perlu kasus uji batasnya sendiri.

Konsep inti bab ini

- IF memilih cabang; urutan ELSIF memengaruhi keputusan.
- LOOP perlu jalan keluar; WHILE dapat berjalan nol kali.
- Jangan menelan error tak dikenal dengan WHEN OTHERS tanpa RAISE.

Urutan pembahasan

- 01 IF, ELSIF dan nested IF
- 02 CASE untuk pilihan yang mudah dibaca
- 03 LOOP, WHILE, FOR dan EXIT
- 04 Exception bawaan dan alur pemulihan
- 05 Custom exception dan RAISE
- 06 RAISE_APPLICATION_ERROR dan pengujian batas

Gambaran konsep dalam satu tabel

Struktur	Cocok untuk	Yang diuji
IF / ELSIF / CASE	Keputusan dan pilihan	NULL, ambang, cabang fallback
LOOP + EXIT	Berhenti dari dalam badan	Jalan keluar yang dapat dicapai
WHILE	Pengulangan berbasis keadaan	Kondisi awal dan perubahan pembatas
FOR	Jumlah iterasi diketahui	Batas dan indeks read-only
EXCEPTION / RAISE	Kegagalan teknis/bisnis	Kode error yang diharapkan

01 IF, ELSIF dan nested IF

IF mengeksekusi cabang THEN hanya ketika kondisi bernilai TRUE.

`FALSE` dan `NULL` tidak menjalankan cabang itu, sehingga data opsional harus ditangani dengan `IS NULL` ketika diperlukan. `ELSIF` mengevaluasi alternatif berurutan; setelah cabang pertama cocok, cabang berikutnya tidak dipilih.

POIN PEMBAHASAN

- Urutan ambang harus dirancang agar kondisi lebih umum tidak menutup kondisi khusus.
- Nested IF berarti keputusan kedua berada di dalam keputusan pertama, misalnya stok cukup baru diikuti keputusan status transaksi.
- Gunakan `END IF` pada tiap tingkat dan indentasi konsisten.
- Latihan memakai variabel lokal supaya percabangan dapat diuji tanpa mengubah data penjualan.
- Bedakan label keluaran dari bukti bahwa transaksi telah tersimpan.

02 CASE untuk pilihan yang mudah dibaca

Simple `CASE` membandingkan sebuah ekspresi dengan beberapa nilai `WHEN`, cocok untuk status `BARU`, `LUNAS` atau `BATAL`.

Searched `CASE` menuliskan kondisi pada setiap `WHEN`, cocok untuk rentang stok. Sebagai statement PL/SQL, setiap cabang berisi tindakan dan ditutup `END CASE`; ini berbeda dari ekspresi `CASE` dalam `SELECT` yang mengembalikan satu nilai.

POIN PEMBAHASAN

- Sediakan `ELSE` untuk masukan yang belum dikenali.
- `CASE` statement tanpa cabang cocok dan tanpa `ELSE` dapat memicu `CASE_NOT_FOUND`, sedangkan `CASE` expression tanpa `ELSE` menghasilkan `NULL` ketika tidak ada kecocokan.
- Pengujian harus mencakup kondisi normal, batas ambang, dan nilai kosong agar label laporan tidak menyembunyikan data yang belum lengkap.

03 LOOP, WHILE, FOR dan EXIT

Basic **LOOP** selalu masuk ke badan loop dan memerlukan jalan keluar.

EXIT meninggalkan loop, sedangkan **EXIT WHEN** meninggalkannya hanya saat kondisi **TRUE**. **WHILE** memeriksa kondisi sebelum badan, sehingga dapat berjalan nol kali.

POIN PEMBAHASAN

- Numeric FOR mengelola indeks lokal dari batas awal sampai akhir; indeks bersifat read-only dan tidak perlu dinaikkan manual.
- Gunakan FOR ketika jumlah iterasi diketahui, **WHILE** ketika bergantung pada keadaan, dan **LOOP** ketika keputusan berhenti berada di badan.
- Pastikan variabel pembatas benar-benar berubah agar tidak membuat infinite loop.
- Contoh berikut sengaja dibatasi kecil.
- Jika kebutuhan hanya agregasi seluruh detail, satu **SELECT SUM** umumnya lebih jelas daripada menjumlahkan dengan loop per baris.

04 Exception bawaan dan alur pemulihan

NO_DATA_FOUND terjadi ketika **SELECT INTO** biasa tidak memperoleh baris, **TOO_MANY_ROWS** ketika memperoleh lebih dari satu, dan **ZERO_DIVIDE** ketika membagi angka dengan nol.

Handler spesifik menerangkan penyebab dan tindakan yang cocok. Setelah exception muncul, sisa pernyataan dalam bagian eksekusi blok itu tidak berjalan; sesudah handler, kontrol melanjutkan di luar blok tersebut.

POIN PEMBAHASAN

- Subblok memungkinkan percobaan yang gagal dipisahkan dari percobaan berikutnya.
- Jangan mencetak berhasil setelah error tanpa menjelaskan batas tindakan yang selesai.
- **WHEN OTHERS** bukan pengganti semua handler: jika hanya mencatat masalah tak dikenal, lakukan **RAISE** agar pemanggil tahu bahwa operasi gagal.

- Contoh tidak memakai DML sehingga tidak memerlukan rollback.

05 Custom exception dan RAISE

Exception yang didefinisikan sendiri dideklarasikan dengan tipe **EXCEPTION** lalu diaktifkan secara eksplisit memakai **RAISE nama_exception**.

Gunakan untuk kondisi bisnis, misalnya jumlah diminta melebihi stok yang tersedia. Handler dapat menampilkan alasan dalam latihan, tetapi pada API yang dipakai program lain, kesalahan biasanya harus dikembalikan kepada pemanggil dengan kontrak yang jelas.

POIN PEMBAHASAN

- Custom exception lokal tidak otomatis mempunyai pesan bisnis publik atau nomor kesalahan khusus.
- Validasi harus dilakukan sebelum DML agar tidak meninggalkan perubahan sebagian.
- Pemilik transaksi menentukan **COMMIT** atau **ROLLBACK**; handler tidak boleh sembarangan membatalkan pekerjaan pemanggil yang tidak terkait.
- Exception saja juga tidak menggantikan constraint maupun mekanisme penguncian untuk sistem serentak.

06 RAISE_APPLICATION_ERROR dan pengujian batas

RAISE_APPLICATION_ERROR mengirim nomor kesalahan aplikasi dalam rentang -20000 sampai -20999 beserta pesan.

Contoh memvalidasi harga sebagai **NUMBER** tanpa skala lebih dahulu: tidak boleh **NULL**, harus positif, dan harus sama dengan **ROUND(harga, 2)**. Deklarasi berskala sejak awal dapat membulatkan input sebelum diuji sehingga kelebihan desimal tidak terlihat.

POIN PEMBAHASAN

- Subblok uji menangkap kesalahan yang diharapkan dengan **SQLCODE** dan mencetak **SQLERRM**; error berbeda dinaikkan lagi.
- Untuk melihat kegagalan diteruskan ke client, jalankan contoh yang tidak mempunyai handler lokal secara terpisah.

- Uji nilai nol, negatif, `NULL`, dua desimal dan desimal berlebih.
- Kerjakan pada XE atau FreeSQL schema pribadi, catat keluaran aktual, dan jangan menyebut contoh tervalidasi Oracle jika belum benar-benar dijalankan.

Alur menyusun kontrol program

- 1 Tentukan keputusan
- 2 Pilih IF atau `CASE`
- 3 Pilih pola loop bila perlu
- 4 Definisikan handler yang tepat
- 5 Uji normal, batas dan gagal

Gunakan alur ini sebagai panduan saat mengikuti praktikum; bukti hasil tetap berasal dari eksekusi Oracle Anda.

AMATI DATA · IKUTI LANGKAH · PAHAM HASIL

Contoh dan tabel pembahasan

Contoh berikut dibaca langsung, tanpa parameter atau tombol simulasi. Nilai adalah **prediksi ilustratif dari data acuan**, bukan hasil eksekusi pada akun Oracle Anda.

Melacak numeric FOR, bukan menebak hasil akhir

`v_total` dimulai dari 0. Setiap iterasi menambahkan nilai `i` ke accumulator.

Contoh SQL / PL/SQL

```
DECLARE
  v_total PLS_INTEGER := 0;
BEGIN
  FOR i IN 1..5 LOOP
```

```

    v_total := v_total + i;
    DBMS_OUTPUT.PUT_LINE(i || ' -> ' || v_total);
END LOOP;
END;
/

```

Iterasi i	v_total setelah penambahan
1	1
2	3
3	6
4	10
5	15

Cara membaca contoh

1. FOR i IN 1..5 memberikan lima nilai indeks pada contoh ini.
2. Perbarui accumulator setelah membaca nilai indeks.
3. Bedakan alur loop normal dari contoh exception; `ZERO_DIVIDE` harus memiliki pembagian dengan nol sebagai penyebabnya.

Yang perlu dipahami: Jejak langkah membantu menemukan kesalahan off-by-one dan penempatan assignment.

Video pembelajaran

Video utama — fokus bab ini

IF/ELSE, CASE dan loop Oracle PL/SQL; referensi kedua melengkapi exception handling.

Oracle PLSQL - IF Else, CASE, Loops

Ora Trainings · [Buka video utama di YouTube](#)

Video tambahan sesuai subtopik

1. [Oracle PL/SQL Tutorial Day -25 ▯ Exception handling in PLSQL▯](#)

NIC IT ACADEMY

Fokus: Exception handling dalam Oracle PL/SQL.

Batas cakupan dan panduan menonton: Menggantikan video CASE saja. Custom exception dan RAISE_APPLICATION_ERROR tetap mengikuti contoh serta pengujian pada halaman.

Video merupakan pendamping, bukan pengganti keseluruhan bab. Pemeriksaan relevansi memakai transkrip yang tersedia, deskripsi pembuat dan bab waktu; bukan klaim seluruh durasi telah ditonton. Versi/editor atau bahasa dapat berbeda. Praktikum tetap mengikuti SQL/PLSQL Oracle pada halaman dan pilihan XE atau FreeSQL login. Jika pemutar dibatasi oleh YouTube, gunakan tautan Buka video.

Praktikum Oracle XE atau FreeSQL

Pilih salah satu lingkungan praktikum

Oracle XE: XE lokal dengan SQL Developer: cocok untuk instalasi, koneksi, user/schema, privilege, tablespace serta latihan SQL/PLSQL.

FreeSQL: FreeSQL dengan login: masuk menggunakan akun sendiri dan gunakan schema latihan yang diizinkan untuk SQL/PLSQL. Tidak perlu memasang XE untuk menyelesaikan latihan non-administratif.

Pengumpulan tugas menerima hasil dari salah satu lingkungan. Pastikan objek prasyarat tersedia dan tetap gunakan schema latihan, bukan schema produksi.

Gunakan [FreeSQL dengan login](#) atau SQL Developer yang terhubung ke XE. [Baca panduan dua jalur praktikum](#). Jalankan kode sesuai urutan dan keterangan tiap langkah.

Output contoh adalah prediksi berdasarkan dataset, bukan bukti bahwa kode telah dijalankan pada akun Anda. Periksa koneksi, hak membuat objek, autocommit dan panel DBMS_OUTPUT. Slash pada baris sendiri adalah pemisah mode skrip klien tertentu, bukan statement PL/SQL.

Gunakan tabel pertemuan02 dan data pertemuan03 atau setup index sekali. Periksa keadaan data bila latihan sebelumnya pernah di-commit. Jangan menjalankan ulang CREATE TABLE/INSERT atau menghapus data secara buta.

Langkah 1 — IF, ELSIF dan nested IF

```
DECLARE
  v_stok NUMBER := 20;
  v_jumlah NUMBER := 2;
  v_status VARCHAR2(12) := 'BARU';
BEGIN
  IF v_jumlah IS NULL OR v_jumlah <= 0 THEN
    DBMS_OUTPUT.PUT_LINE('Jumlah tidak valid');
  ELSIF v_stok IS NULL THEN
    DBMS_OUTPUT.PUT_LINE('Stok belum diketahui');
  ELSIF v_jumlah > v_stok THEN
    DBMS_OUTPUT.PUT_LINE('Stok kurang');
  ELSE
    IF v_status = 'BARU' THEN
      DBMS_OUTPUT.PUT_LINE('Siap diperiksa');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Status bukan BARU');
    END IF;
  END IF;
END;
/
```

Ubah nilai variabel untuk mencapai semua cabang. Kondisi NULL ditangani eksplisit, tidak diasumsikan TRUE atau FALSE.

Langkah 2 — Simple CASE dan searched CASE

```
DECLARE
  v_status VARCHAR2(12) := 'LUNAS';
  v_stok NUMBER := 0;
BEGIN
  CASE v_status
    WHEN 'BARU' THEN DBMS_OUTPUT.PUT_LINE('Menunggu proses');
    WHEN 'LUNAS' THEN DBMS_OUTPUT.PUT_LINE('Pembayaran tercatat');
    WHEN 'BATAL' THEN DBMS_OUTPUT.PUT_LINE('Dibatalkan');
```

```

        ELSE DBMS_OUTPUT.PUT_LINE('Status tidak dikenali');
    END CASE;
CASE
    WHEN v_stok IS NULL THEN DBMS_OUTPUT.PUT_LINE('Stok tidak
diketahui');
    WHEN v_stok = 0 THEN DBMS_OUTPUT.PUT_LINE('Stok kosong');
    WHEN v_stok < 10 THEN DBMS_OUTPUT.PUT_LINE('Stok rendah');
    ELSE DBMS_OUTPUT.PUT_LINE('Stok mencukupi');
END CASE;
END;
/

```

Keduanya adalah CASE statement PL/SQL dan memakai END CASE, bukan sekadar ekspresi CASE pada SELECT.

Langkah 3 — Basic LOOP dengan EXIT WHEN dan EXIT

```

DECLARE
    v_i PLS_INTEGER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE('LOOP i=' || v_i);
        EXIT WHEN v_i >= 3;
        v_i := v_i + 1;
    END LOOP;
    v_i := 1;
    LOOP
        IF v_i > 2 THEN
            EXIT;
        END IF;
        DBMS_OUTPUT.PUT_LINE('EXIT biasa i=' || v_i);
        v_i := v_i + 1;
    END LOOP;
END;
/

```

Batas dan perubahan penghitung terlihat jelas. EXIT biasa di dalam IF berbeda sintaks dengan EXIT WHEN.

Langkah 4 — WHILE LOOP dan numeric FOR LOOP

```

DECLARE
  v_i PLS_INTEGER := 1;
BEGIN
  WHILE v_i <= 3 LOOP
    DBMS_OUTPUT.PUT_LINE('WHILE i=' || v_i);
    v_i := v_i + 1;
  END LOOP;
  FOR k IN 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE('FOR k=' || k);
  END LOOP;
  FOR k IN REVERSE 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE('FOR turun k=' || k);
  END LOOP;
END;
/

```

WHILE memeriksa sebelum badan; FOR menaikkan indeks otomatis dan REVERSE menurunkannya. Jangan assign ke indeks k.

Langkah 5 — Tiga exception bawaan dalam subblok

```

DECLARE
  v_nama pelanggan_demo.nama%TYPE;
  v_hasil NUMBER;
  v_pembagi NUMBER := 0;
BEGIN
  BEGIN
    SELECT nama INTO v_nama FROM pelanggan_demo WHERE 1=0;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Tidak ada
    pelanggan cocok');
  END;
  BEGIN
    SELECT nama INTO v_nama FROM pelanggan_demo;
    DBMS_OUTPUT.PUT_LINE('Hanya satu pelanggan pada schema ini');
  EXCEPTION
    WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Lebih dari satu
    pelanggan');
  END;
END;

```

```

        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Tabel pelanggan
kosong');
    END;
    BEGIN
        v_hasil := 100 / v_pembagi;
        DBMS_OUTPUT.PUT_LINE('Baris ini tidak dijalankan jika pembagi
nol');
    EXCEPTION
        WHEN ZERO_DIVIDE THEN DBMS_OUTPUT.PUT_LINE('Pembagian dengan
nol ditolak');
    END;
    DBMS_OUTPUT.PUT_LINE('Percobaan selesai pada blok luar');
END;
/

```

Seed menjamin kasus banyak pelanggan; query pertama selalu kosong dan pembagi nol selalu memicu ZERO_DIVIDE. Tidak ada klaim output telah dijalankan Oracle.

Langkah 6 — Custom exception dan RAISE eksplisit

```

DECLARE
    e_stok_kurang EXCEPTION;
    v_stok NUMBER := 2;
    v_diminta NUMBER := 3;
BEGIN
    IF v_diminta > v_stok THEN
        RAISE e_stok_kurang;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Stok memenuhi permintaan');
EXCEPTION
    WHEN e_stok_kurang THEN
        DBMS_OUTPUT.PUT_LINE('Permintaan ditolak: stok kurang');
END;
/

```

Handler menangani hanya custom exception ini. Pesan memenuhi permintaan tidak berjalan ketika RAISE dieksekusi.

Langkah 7 — RAISE_APPLICATION_ERROR: uji harga dua

desimal

```

DECLARE
    v_harga NUMBER;
BEGIN
    FOR i IN 1..5 LOOP
        v_harga := CASE i
            WHEN 1 THEN 0
            WHEN 2 THEN -1
            WHEN 3 THEN NULL
            WHEN 4 THEN 15000.25
            ELSE 15000.251
        END;
        BEGIN
            IF v_harga IS NULL OR v_harga <= 0 OR v_harga <>
ROUND(v_harga,2) THEN
                RAISE_APPLICATION_ERROR(-20001, 'Harga wajib positif dan
maksimal dua desimal');
            END IF;
            DBMS_OUTPUT.PUT_LINE('Valid: ' || v_harga);
        EXCEPTION
            WHEN OTHERS THEN
                IF SQLCODE = -20001 THEN
                    DBMS_OUTPUT.PUT_LINE('Ditolak: ' || SQLERRM);
                ELSE
                    RAISE;
                END IF;
            END;
        END LOOP;
    END;
/

```

NUMBER tanpa skala mempertahankan masukan desimal berlebih sampai validasi. Kasus yang valid tetap terlihat; handler menaikkan ulang error tak terduga, tanpa COMMIT atau ROLLBACK.

Langkah 8 — Kesalahan aplikasi diteruskan ke client

```

BEGIN
    RAISE_APPLICATION_ERROR(-20002, 'Contoh kegagalan yang

```

```
diteruskan ke pemanggil');
END;
/
```

Jalankan terpisah: error aplikasi memang diharapkan karena tidak ada handler lokal. Catat pesan aktual; jangan menganggap runner berhenti sebagai kesalahan bahan ajar.

Tugas / latihan

1. Pada XE ATAU FreeSQL schema pribadi, uji semua cabang IF/ELSIF/nested IF dan dua CASE; buat tabel kondisi input dan output aktual.
2. Tulis LOOP, WHILE dan FOR yang dibatasi jelas; gunakan EXIT serta EXIT WHEN, lalu jelaskan alasan loop berhenti.
3. Jalankan uji tiga exception bawaan, custom exception dan validasi RAISE_APPLICATION_ERROR. Sertakan kasus harga NULL, nol, negatif, dua desimal dan lebih dari dua desimal; tidak perlu mengubah tabel.

Pertanyaan analisis

1. Mengapa handler yang hanya mencetak berhasil lalu mengabaikan WHEN OTHERS berbahaya bagi pemanggil? Jelaskan kapan RAISE kembali diperlukan.
2. Mengapa NUMBER(12,2) pada variabel masukan dapat merusak uji deteksi desimal berlebih, dan bagaimana kontrak transaksi harus tetap dimiliki pemanggil?

Pengumpulan

Tugas boleh dikerjakan di **Oracle XE lokal atau FreeSQL dengan login**; cukup pilih salah satu. Kumpulkan satu laporan melalui LMS/kanal yang ditetapkan dosen. Cantumkan Nama, NPM, pertemuan dan lingkungan yang dipakai. Sertakan kode SQL/PLSQL sebagai teks, output aktual atau pesan error, serta jawaban analisis.

Jika FreeSQL menyediakan tautan berbagi yang dapat diakses dosen, tautan boleh ditambahkan sebagai bukti pendamping, bukan pengganti hasil eksekusi yang terdokumentasi. Jangan membagikan password, token atau koneksi pribadi.

Hasil yang belum dieksekusi diberi label “prediksi”. Screenshot ilustrasi browser bukan bukti praktikum Oracle.

Rubrik penilaian

- Ketepatan konsep dan rancangan: **20%**
 - Kebenaran query/program: **30%**
 - Pengujian normal, batas dan gagal: **20%**
 - Analisis serta penjelasan hasil: **20%**
 - Identitas dan kerapian laporan: **10%**
-

Referensi resmi

- [Oracle 19c — PL/SQL Control Statements](#)
- [Oracle 19c — PL/SQL Error Handling](#)
- [Oracle 19c — SELECT INTO](#)

Panduan XE21c digunakan untuk konteks instalasi lokal; dokumentasi SQL/PLSQL untuk konsep bahasa. Versi FreeSQL/editor dapat berbeda, sehingga periksa fitur dan pesan aktual.
