

ORACLE PL/SQL · PERTEMUAN 12 / 14

Procedure & Function

Kontrak parameter, perhitungan murni, dan transaksi yang dimiliki pemanggil.

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

Versi PDF materi pertemuan 12. Contoh dan tabel dapat dibaca di PDF. Kuis dapat dijalankan di halaman web: [Buka materi daring](#).

Tujuan pembelajaran

1. Membuat dan memanggil CREATE OR REPLACE PROCEDURE.
2. Membedakan parameter IN, OUT, serta IN OUT.
3. Membangun fungsi RETURN yang bisa dipakai dalam SELECT.
4. Memvalidasi jumlah integer positif dan harga positif maksimal dua desimal.
5. Menguji INSERT/UPDATE dengan ID eksplisit dan rollback pemanggil.

Pokok bahasan / praktikum

- CREATE OR REPLACE PROCEDURE
- Parameter IN OUT IN OUT
- Pemanggilan positional dan named notation
- INSERT UPDATE dan transaction owner
- CREATE OR REPLACE FUNCTION RETURN
- Function dalam SQL
- Procedure vs function
- Kasus perhitungan jumlah dan harga
- USER_ERRORS

Prasyarat: Modul 02/03, blok PL/SQL dan exception, serta transaksi modul 11.
Sequence belum diperlukan: masukkan ID eksplisit yang belum dipakai.

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. Apakah subprogram menghitung total harus sekaligus menyimpan transaksi?
2. Bagaimana procedure mengembalikan lebih dari satu informasi?
3. Apa risikonya bila helper melakukan COMMIT sendiri?

Uji pemahaman awal

1. Mode parameter yang menerima nilai awal dan mengirim nilai baru adalah?

- IN
- OUT
- IN OUT
- RETURN

2. Subprogram yang paling tepat untuk SELECT menghitung subtotal tanpa efek samping?

- Procedure dengan COMMIT
- Function RETURN NUMBER tanpa DML
- Trigger INSERT
- Cursor terbuka global

3. Mengapa procedure latihan tidak melakukan COMMIT?

- COMMIT tidak tersedia di Oracle
- Agar OUT selalu NULL
- Agar pemanggil menentukan atomisitas seluruh transaksi
- Karena setiap INSERT sudah permanen

PAHAMI KONSEP · BACA TERSTRUKTUR · TERAPKAN

Teori dan pembahasan

Definisikan kontrak subprogram sebelum menulis implementasinya. Pisahkan tindakan pengubah data dari fungsi perhitungan yang dapat diuji tanpa efek samping.

Konsep inti bab ini

- Procedure cocok untuk tindakan; function mengembalikan nilai.
- OUT/ IN OUT membutuhkan variabel pemanggil yang sesuai.
- Pisahkan DDL kompilasi dari transaksi pengujian; periksa USER_ERRORS .

Urutan pembahasan

- 01 Kontrak subprogram dan lingkungan
- 02 Parameter IN, OUT, dan IN OUT
- 03 Function RETURN dan validasi
- 04 Pemakaian function di SQL
- 05 INSERT dan UPDATE sebagai satu pekerjaan
- 06 Pengujian dan diagnosis

Gambaran konsep dalam satu tabel

Kontrak	Tujuan	Contoh
Procedure	Menjalankan tindakan	<code>INSERT</code> header/detail dan <code>UPDATE</code> stok
Function <code>RETURN</code>	Menghasilkan nilai	Subtotal jumlah × harga
IN	Masukan	Jumlah dan harga
OUT	Keluaran	Total atau ID hasil
IN OUT	Masukan dan keluaran	Counter panggilan

01 Kontrak subprogram dan lingkungan

Gunakan Oracle XE 21c lokal melalui SQL Developer ATAU Oracle FreeSQL setelah login pada schema sendiri yang diizinkan; bukan dua kewajiban.

Selesaikan setup lima tabel dan seed modul 02/03 sekali, tanpa menghapus data. Aktifkan panel DBMS Output pada koneksi lokal atau output editor FreeSQL. SQL memakai titik koma; slash pada baris sendiri adalah delimiter Run Script klien, bukan sintaks server. Jika editor mengirim satu unit PL/SQL, kirim blok tanpa slash. Seluruh latihan menggunakan objek sendiri, tanpa DBA atau langkah administrasi.

POIN PEMBAHASAN

- Subprogram tersimpan memberi nama pada pekerjaan berulang.
- Procedure cocok untuk tindakan, misalnya membuat penjualan; function cocok untuk nilai, misalnya menghitung total.
- `CREATE OR REPLACE` menyimpan definisi pada schema saat ini dan termasuk DDL.
- Jalankan tahap pembuatan sebelum transaksi pengujian, bukan di tengah perubahan yang ingin dibatalkan.
- Sukses mengirim `CREATE` tidak menjamin objek valid; lihat `USER_ERRORS` dan status objek.

02 Parameter IN, OUT, dan IN OUT

IN adalah masukan dan menjadi mode default.

Procedure membaca nilainya tanpa menjadikannya jalur hasil. OUT mengirim hasil ke variabel pemanggil, sehingga jangan mengirim literal sebagai argumen OUT. **IN OUT** menerima nilai awal sekaligus mengembalikan nilai baru; perubahan variabel PL/SQL ini tidak dibatalkan hanya karena tabel di-rollback.

POIN PEMBAHASAN

- Gunakan nama parameter dengan awalan p_ dan variabel lokal v_.
- Contoh parameter p_jumlah, p_harga, p_total OUT, serta p_panggilan **IN OUT** memisahkan data bisnis dari pencatatan pemanggilan.
- Nilai OUT jangan dipercaya sesudah panggilan gagal.
- Positional notation mengikuti urutan deklarasi; named notation seperti p_harga => 30000 membuat makna panggilan lebih mudah diaudit.

03 Function RETURN dan validasi

Function mendeklarasikan **RETURN NUMBER** dan setiap jalur normal wajib menghasilkan nilai melalui **RETURN**.

Fungsi perhitungan menerima jumlah integer positif, bukan **NULL** atau pecahan; harga wajib positif dan sama dengan ROUND(harga,2). Validasi dilakukan sebelum perkalian agar harga tiga desimal tidak diam-diam dibulatkan oleh kolom **NUMBER(12,2)**.

RAISE_APPLICATION_ERROR menyampaikan kegagalan yang dapat dikenali pemanggil.

POIN PEMBAHASAN

- Harga kolom produk sudah disimpan dua desimal; input fungsi tetap harus diperiksa karena dapat berasal dari literal atau variabel luar.
- Jumlah juga dibatasi maksimum 999999 agar sesuai kolom jumlah **NUMBER(6)**.
- Harga dibatasi kapasitas kolom jika akan dipakai menyimpan detail.

- Aturan batas penyimpanan melengkapi, bukan menggantikan, pemeriksaan positif dan integer.

04 Pemakaian function di SQL

SELECT `fn_total_demo(jumlah,harga)` membaca nilai untuk tiap detail tanpa menyisipkan penjualan baru.

Function yang dipanggil SQL di latihan ini hanya mempunyai parameter IN, tidak melakukan DML, dan tidak mengendalikan transaksi. Dengan seed, total sale 101 berasal dari sepuluh Kopi pada harga tersimpan, bukan selalu harga terbaru produk.

POIN PEMBAHASAN

- Procedure dipanggil sebagai statement PL/SQL, bukan ekspresi `SELECT`.
- Function dapat dipanggil dalam assignment PL/SQL maupun ekspresi SQL apabila kontraknya memenuhi aturan SQL.
- Jangan menandai fungsi DETERMINISTIC sekadar untuk menghilangkan pesan kesalahan.
- Pisahkan fungsi murni dan procedure pengubah tabel agar perhitungan bisa diuji tanpa efek samping.

05 INSERT dan UPDATE sebagai satu pekerjaan

Procedure `pr_buat_jual_demo` menerima ID eksplisit yang belum digunakan karena **sequence baru diperkenalkan modul 13**.

Pelanggan harus ditemukan, produk dibaca dengan `FOR UPDATE`, stok diperiksa, lalu header serta detail dimasukkan dan stok dikurangi. Lock mencegah dua transaksi mengandalkan stok lama secara bersamaan. ID bentrok menghasilkan exception, bukan alasan menimpa penjualan lama.

POIN PEMBAHASAN

- Procedure ini tidak melakukan `COMMIT` atau `ROLLBACK` penuh.
- Pemanggil memilih apakah keseluruhan pekerjaan disimpan.

- Dalam contoh mandiri, exception menyebabkan `ROLLBACK` oleh pemanggil yang memang memiliki seluruh transaksi.
- Pada aplikasi besar, desain batas transaksi harus dinyatakan; menangkap exception lalu meneruskan tanpa rollback dapat meninggalkan DML sebelumnya sebagai pekerjaan tertunda.

06 Pengujian dan diagnosis

Jalankan helper parameter terlebih dahulu, lalu fungsi dalam SQL, kemudian procedure penjualan dengan ID 190 bila masih kosong.

Uji `NULL`, jumlah pecahan, harga tiga desimal, pelanggan tidak ada, serta stok kurang.

Tampilkan pesan yang benar-benar muncul dan kembalikan seed dengan rollback.

`DBMS_OUTPUT` hanya pesan, bukan bukti data tersimpan.

POIN PEMBAHASAN

- Inspect `USER_ERRORS` sesudah definisi function atau procedure.
- Bila ada kesalahan, perbaiki definisi di tahap DDL terpisah; jangan menyebut objek sudah terkompilasi sebelum memeriksanya pada Oracle.
- `NULL` dan string kosong perlu perhatian: pelanggan anonim pada seed sah menurut tabel, tetapi procedure penjualan baru sengaja mewajibkan pelanggan terdaftar.
- Kontrak bisnis dapat lebih ketat daripada nullability tabel.

Alur membangun subprogram

- 1 Nyatakan parameter/hasil
- 2 Validasi input
- 3 Buat definisi dan cek `USER_ERRORS`
- 4 Panggil dari SQL/PLSQL sesuai kontrak
- 5 Uji lalu putuskan transaksi

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.

Function menghasilkan total; procedure mempunyai parameter keluaran

Dengan jumlah 2 dan harga 30000, perhitungan total dapat dipahami sebelum memanggil subprogram.

Contoh SQL / PL/SQL

```
-- Setelah fn_total_demo berhasil dibuat:
SELECT fn_total_demo(2,30000) AS total FROM dual;
```

Komponen	Nilai contoh	Arti
IN p_jumlah	2	Masukan jumlah
IN p_harga	30000	Masukan harga

Komponen	Nilai contoh	Arti
RETURN / OUT p_total	60000	Hasil perhitungan
IN OUT counter: sebelum	0	Nilai pemanggil
IN OUT counter: sesudah	1	Nilai setelah satu panggilan

Cara membaca contoh

1. Function `fn_total_demo` mengembalikan jumlah × harga, bukan menyimpan transaksi.
2. Procedure `pr_ringkas_demo` mengisi OUT p_total dan menaikkan counter `IN OUT`.
3. Jalankan definisi function/procedure pada praktikum sebelum memanggilnya.

Yang perlu dipahami: Nilai RETURN/OUT tidak berarti COMMIT. Perubahan database dan kepemilikan transaksi adalah kontrak lain.

Video pembelajaran

Video utama — fokus bab ini

Stored procedure, function RETURN, pemanggilan dan contoh validasi/insert Oracle. Video tambahan membahas IN, OUT dan IN OUT.

Oracle PL/SQL Stored Procedures and Functions

Clint Tuttle · [Buka video utama di YouTube](#)

Video tambahan sesuai subtopik

1. [Procedure in plsql. In, out & In out parameter types explained with simple example.](#)
Swisstac

Fokus: Parameter procedure Oracle PL/SQL: IN, OUT dan IN OUT dengan contoh.

Batas cakupan dan panduan menonton: Procedure dan function dibandingkan melalui praktik, termasuk perhitungan transaksi. Kepemilikan COMMIT/ROLLBACK tetap mengikuti pemanggil pada kode 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 — Function perhitungan yang dapat digunakan SQL

```
CREATE OR REPLACE FUNCTION fn_total_demo(p_jumlah IN NUMBER,
p_harga IN NUMBER)
RETURN NUMBER IS
BEGIN
  IF p_jumlah IS NULL OR p_jumlah <= 0 OR p_jumlah <>
```

```

TRUNC(p_jumlah) OR p_jumlah > 999999 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Jumlah harus integer positif
maksimal 999999');
END IF;
IF p_harga IS NULL OR p_harga <= 0 OR p_harga <> ROUND(p_harga,2)
OR p_harga > 9999999999.99 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Harga harus positif, maksimal
dua desimal dan sesuai kapasitas kolom');
END IF;
RETURN p_jumlah * p_harga;
END;
/
SELECT fn_total_demo(2,30000) AS total FROM dual;
SELECT id_jual, fn_total_demo(jumlah,harga) AS subtotal FROM
detail_jual_demo ORDER BY id_jual;

```

Prediksi 60000 untuk literal; subtotal seed 300000, 150000, 60000. Tidak mengubah tabel.

Langkah 2 — Procedure dengan IN, OUT, IN OUT dan dua cara memanggil

```

CREATE OR REPLACE PROCEDURE pr_ringkas_demo(
    p_jumlah IN NUMBER, p_harga IN NUMBER,
    p_total OUT NUMBER, p_panggilan IN OUT NUMBER
) IS
BEGIN
    IF p_panggilan IS NULL OR p_panggilan < 0 OR p_panggilan <>
TRUNC(p_panggilan) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Counter harus integer
nonnegatif');
    END IF;
    p_total := fn_total_demo(p_jumlah,p_harga);
    p_panggilan := p_panggilan + 1;
END;
/
DECLARE
    v_total NUMBER;
    v_panggilan NUMBER := 0;
BEGIN
    pr_ringkas_demo(2,30000,v_total,v_panggilan);

```

```

    DBMS_OUTPUT.PUT_LINE('Total=' || v_total || ', panggilan=' ||
v_panggilan);
    pr_ringkas_demo(p_jumlah => 3,p_harga => 15000,p_total =>
v_total,p_panggilan => v_panggilan);
    DBMS_OUTPUT.PUT_LINE('Total=' || v_total || ', panggilan=' ||
v_panggilan);
END;
/

```

Counter diprediksi 1 lalu 2; total 60000 lalu 45000. OUT dan IN OUT menggunakan variabel pemanggil.

Langkah 3 — Procedure INSERT header/detail serta UPDATE stok

```

CREATE OR REPLACE PROCEDURE pr_buat_jual_demo(
    p_id IN NUMBER, p_pelanggan IN NUMBER, p_produk IN NUMBER,
    p_jumlah IN NUMBER
) IS
    v_dummy NUMBER;
    v_harga produk_demo.harga%TYPE;
    v_stok produk_demo.stok%TYPE;
    v_total NUMBER;
BEGIN
    IF p_id IS NULL OR p_id <= 0 OR p_id <> TRUNC(p_id) OR p_id >
99999999 THEN
        RAISE_APPLICATION_ERROR(-20004, 'ID penjualan integer positif
maksimal 8 digit');
    END IF;
    BEGIN
        SELECT id_pelanggan INTO v_dummy FROM pelanggan_demo WHERE
id_pelanggan = p_pelanggan;
        EXCEPTION WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20005,
'Pelanggan tidak ditemukan');
    END;
    BEGIN
        SELECT harga,stok INTO v_harga,v_stok FROM produk_demo WHERE
id_produk = p_produk FOR UPDATE;
        EXCEPTION WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20006,
'Produk tidak ditemukan');
    END;

```

```

v_total := fn_total_demo(p_jumlah,v_harga);
IF v_stok < p_jumlah THEN RAISE_APPLICATION_ERROR(-20007, 'Stok
tidak cukup'); END IF;
INSERT INTO penjualan_demo(id_jual,id_pelanggan,tanggal,status)
VALUES(p_id,p_pelanggan,SYSDATE,'BARU');
INSERT INTO detail_jual_demo(id_jual,id_produk,jumlah,harga)
VALUES(p_id,p_produk,p_jumlah,v_harga);
UPDATE produk_demo SET stok = stok - p_jumlah WHERE id_produk =
p_produk;
END;
/

```

Tidak memakai sequence modul 13 dan tidak menyimpan transaksi secara tersembunyi. FOR UPDATE mempertahankan lock hingga pemanggil menyelesaikan transaksi.

Langkah 4 — Pemanggil mandiri sukses tetapi rollback untuk menjaga seed

```

DECLARE
v_ada NUMBER;
BEGIN
SELECT COUNT(*) INTO v_ada FROM penjualan_demo WHERE id_jual =
190;
IF v_ada > 0 THEN RAISE_APPLICATION_ERROR(-20008, 'ID 190
terpakai: pilih ID eksplisit lain'); END IF;
pr_buat_jual_demo(190,1,11,2);
FOR r IN (SELECT id_jual,jumlah,harga FROM detail_jual_demo WHERE
id_jual = 190) LOOP
DBMS_OUTPUT.PUT_LINE('Penjualan=' || r.id_jual || ', jumlah=' ||
r.jumlah);
END LOOP;
ROLLBACK;
EXCEPTION WHEN OTHERS THEN
ROLLBACK;
RAISE;
END;
/
SELECT COUNT(*) AS sisa_uji FROM penjualan_demo WHERE id_jual =
190;
SELECT stok FROM produk_demo WHERE id_produk = 11;

```

Gunakan koneksi tanpa pekerjaan tertunda. Sisa_uji 0 dan stok 20 diprediksi untuk seed. Untuk demonstrasi simpan permanen, ubah ROLLBACK jalur sukses menjadi COMMIT hanya setelah keputusan sadar dan ID baru.

Langkah 5 — Kasus gagal perhitungan dan stok

```

DECLARE
  v_total NUMBER;
BEGIN
  BEGIN
    v_total := fn_total_demo(1.5,30000);
    RAISE_APPLICATION_ERROR(-20990,'Uji gagal: pecahan diterima');
  EXCEPTION WHEN OTHERS THEN
    IF SQLCODE <> -20001 THEN RAISE; END IF;
    DBMS_OUTPUT.PUT_LINE('Ditolak sesuai kontrak: ' || SQLERRM);
  END;
  BEGIN
    v_total := fn_total_demo(1,100.123);
    RAISE_APPLICATION_ERROR(-20990,'Uji gagal: tiga desimal
diterima');
  EXCEPTION WHEN OTHERS THEN
    IF SQLCODE <> -20002 THEN RAISE; END IF;
    DBMS_OUTPUT.PUT_LINE('Ditolak sesuai kontrak: ' || SQLERRM);
  END;
  BEGIN
    pr_buat_jual_demo(191,1,14,1);
    RAISE_APPLICATION_ERROR(-20990,'Uji gagal: stok kosong
diterima');
  EXCEPTION WHEN OTHERS THEN
    IF SQLCODE <> -20007 THEN RAISE; END IF;
    DBMS_OUTPUT.PUT_LINE('Ditolak sesuai kontrak: ' || SQLERRM);
  END;
  ROLLBACK;
EXCEPTION WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

```

Hanya pesan error yang diharapkan dianggap lulus; exception lain dilempar kembali. Pemanggil membatalkan seluruh transaksi uji.

Langkah 6 — Inspeksi kompilasi

```
SELECT name, type, line, position, text
FROM user_errors
WHERE name IN
('FN_TOTAL_DEMO', 'PR_RINGKAS_DEMO', 'PR_BUAT_JUAL_DEMO')
ORDER BY name, type, sequence;
SELECT object_name, status FROM user_objects WHERE object_name IN
('FN_TOTAL_DEMO', 'PR_RINGKAS_DEMO', 'PR_BUAT_JUAL_DEMO');
```

USER_ERRORS kosong dan status VALID adalah bukti yang harus dikumpulkan setelah menjalankan Oracle, bukan klaim verifikasi dari materi ini.

Tugas / latihan

1. Pada XE lokal ATAU FreeSQL milik sendiri, buat fungsi diskon dengan parameter nominal positif maksimal dua desimal; tolak input tidak valid.
2. Panggil procedure parameter melalui positional dan named notation; tampilkan perubahan OUT serta IN OUT.
3. Uji penjualan dengan ID eksplisit kosong, pelanggan/produk tidak ada, dan stok kurang; rollback perubahan pada koneksi latihan.
4. Sertakan definisi lengkap, USER_ERRORS aktual, prediksi versus hasil, serta penjelasan pemilik transaksi.

Pertanyaan analisis

1. Apakah rollback tabel mengembalikan nilai variabel IN OUT? Jelaskan perbedaan memori PL/SQL dan transaksi.
2. Mengapa harga harus diperiksa sebelum masuk kolom NUMBER(12,2), bukan setelah database membulatkannya?

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 Subprograms](#)
- [Oracle 19c — COMMIT](#)
- [Oracle 19c — SELECT, FOR UPDATE](#)
- [Oracle 19c — USER_ERRORS](#)

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