

## ORACLE PL/SQL · PERTEMUAN 14 / 14

# Final Project Oracle & PL/SQL

Satu penjualan, satu kontrak bisnis, satu keputusan transaksi.

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

Versi PDF materi pertemuan 14. Contoh dan tabel dapat dibaca di PDF. Kuis dapat dijalankan di halaman web: [Buka materi daring](#).

---

## Tujuan pembelajaran

1. Merangkai lima tabel relational beserta constraint dan dataset bersama.
2. Mengintegrasikan sequence, view, trigger audit, procedure, function, cursor, exception, transaksi, dan package.
3. Membangun pkg\_penjualan\_demo lengkap dengan validasi dan lock stok FOR UPDATE.
4. Menguji jalur sukses/gagal, rollback, dan batas kompilasi Oracle secara jujur.
5. Mempresentasikan ERD, kontrak API, bukti data, serta keterbatasan proyek latihan.

## Pokok bahasan / praktikum

- Lima tabel relational dan constraints
- Sequence seq\_penjualan\_demo
- View v\_total\_penjualan\_demo
- Trigger audit\_stok\_demo trg\_audit\_stok\_demo
- Package spec/body pkg\_penjualan\_demo
- Function hitung\_total(p\_jumlah,p\_harga) RETURN NUMBER
- Procedure proses(p\_pelanggan,p\_produk,p\_jumlah,p\_id OUT NUMBER)
- Validasi integer positif dan harga maksimal dua desimal
- FOR UPDATE stock lock
- Cursor FOR procedure lapor\_stok
- Exception dan RAISE\_APPLICATION\_ERROR
- Transaction owner COMMIT ROLLBACK SAVEPOINT
- Pengujian USER\_ERRORS dan presentasi

**Prasyarat:** Selesaikan modul 02/03 dan 11–13. Wajib memakai seq\_penjualan\_demo, v\_total\_penjualan\_demo, audit\_stok\_demo, trg\_audit\_stok\_demo yang telah dibuat modul 13; definisinya tersedia penuh di modul 13. Jangan membuat ulang/reset sequence atau menghapus data. pkg\_latihan\_demo bukan package final.

### 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. Objek mana yang bertanggung jawab mengalokasikan ID, menghitung total, dan menyimpan audit?
2. Apa yang terjadi ketika stok telah dibaca tetapi transaksi lain mencoba membeli produk sama?
3. Bukti apa yang membedakan pesan sukses dari transaksi yang benar-benar konsisten?

## Uji pemahaman awal

### 1. Urutan aman untuk mengurangi stok dalam proses adalah?

- Baca stok tanpa lock lalu commit audit
- Lock produk FOR UPDATE, validasi stok, lakukan DML, pemanggil menyelesaikan transaksi
- Ambil MAX(id)+1 lalu hapus produk
- Commit sebelum detail dimasukkan

### 2. Setelah uji sukses di-rollback, apa yang diharapkan?

- Nomor sequence kembali 200
- Header tetap ada tetapi detail hilang
- Stok, header/detail, dan audit kembali; nomor sequence boleh berlubang
- Trigger dinonaktifkan otomatis

### 3. Bukti terkuat bahwa package dapat dipakai adalah?

- JSON valid saja
- Teks CREATE terlihat benar
- DBMS\_OUTPUT mencetak judul
- USER\_ERRORS dan status objek aktual, disertai uji Oracle sukses/gagal

**PAHAMI KONSEP · BACA TERSTRUKTUR · TERAPKAN**

## Teori dan pembahasan

Integrasikan seluruh komponen menjadi satu unit kerja penjualan. Tampilkan bukan hanya jalur sukses, tetapi juga pemulihan data dan batas solusi.

## Konsep inti bab ini

- Validasi dan FOR UPDATE mendahului perubahan header/detail serta stok.
- Package tidak melakukan COMMIT / ROLLBACK penuh secara tersembunyi.
- Uji rollback harus memulihkan data dan audit; nomor sequence boleh berlubang.

## Urutan pembahasan

- 01 Ruang lingkup dan alur proyek
- 02 Relasi dan constraint sebagai fondasi
- 03 Dependensi modul 13 dan tahap DDL
- 04 Function hitung\_total sebagai kontrak perhitungan
- 05 Procedure proses dan stock lock
- 06 Audit, cursor, dan pemilik transaksi
- 07 Matriks uji dan bukti yang dapat diperiksa
- 08 Presentasi dan batas verifikasi

## Gambaran konsep dalam satu tabel

| Lapisan             | Komponen                   | Tujuan pemeriksaan              |
|---------------------|----------------------------|---------------------------------|
| Data                | Tabel dan constraints      | Integritas hubungan dan nilai   |
| Identitas / laporan | Sequence dan view          | Nomor unik serta total historis |
| Logika bisnis       | Procedure/function/package | Kontrak dan validasi            |
| Respons / pembacaan | Trigger dan cursor         | Audit serta laporan stok        |
| Pengujian           | Exception dan transaction  | Sukses, gagal dan pemulihan     |

## 01 Ruang lingkup dan alur proyek

**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

- Proyek berupa penjualan kecil dengan satu produk per panggilan.
- Ini latihan integrasi, bukan sistem pembayaran atau persediaan siap produksi.
- Diagram alurnya adalah pelanggan dan produk diperiksa, produk dikunci, jumlah/harga divalidasi, stok dibandingkan, ID sequence diambil, header/detail dimasukkan, stok dikurangi, dan audit otomatis dicatat.
- Pemanggil kemudian memutuskan `COMMIT` atau `ROLLBACK`.
- Semua langkah DML harus diperlakukan sebagai satu unit kerja.

## 02 Relasi dan constraint sebagai fondasi

**Gunakan kelima tabel bersama, bukan mengganti dataset dengan tabel baru yang mirip. `pelanggan_demo` memiliki primary key serta email `UNIQUE`; `produk_demo` mempunyai harga positif dan stok nonnegatif; `penjualan_demo` mengacu pelanggan dan membatasi status; `detail_jual_demo` memiliki composite primary key serta dua foreign key; `pegawai_demo` mempunyai self-reference manajer.**

Tabel pegawai digunakan untuk laporan penanggung jawab dan demonstrasi hierarki, tidak ditambahkan diam-diam sebagai kolom penjualan.

### POIN PEMBAHASAN

- Seed mencakup pelanggan tanpa penjualan, transaksi anonim dengan pelanggan `NULL`, produk tanpa detail, serta pohon manajer.

- Laporan LEFT JOIN mempertahankan pelanggan tanpa penjualan.
- API penjualan baru lebih ketat daripada seed: pelanggan wajib terdaftar dan tidak `NULL`.
- Constraint tetap menjaga integritas bila pemanggil melewati API, tetapi validasi procedure memberikan pesan bisnis yang lebih mudah dimengerti.

### 03 Dependensi modul 13 dan tahap DDL

**Sequence, aggregate view, audit table, dan audit trigger harus sudah tersedia dari modul 13.**

Inventaris `USER_OBJECTS` dan `USER_SEQUENCES` menunjukkan keberadaan, status, dan increment; `USER_ERRORS` menunjukkan masalah kompilasi. Nilai `LAST_NUMBER` bukan jaminan ID berikutnya untuk semua konfigurasi sequence. Jangan me-reset sequence hanya karena tes sebelumnya mengonsumsi nomor 200.

#### POIN PEMBAHASAN

- Package final mempunyai nama tetap `pkg_penjualan_demo`.
- Specification dipublikasikan terlebih dahulu dan body mengimplementasikan ketiga anggota.
- DDL package dapat commit pekerjaan sebelumnya, sehingga tahap kompilasi selesai sebelum pengujian rollback. `pkg_latihan_demo` tetap independen.
- Kode tidak menyembunyikan body, import paket eksternal, atau langkah wajib yang hanya dijelaskan secara verbal; anggota publik dan implementasinya ditampilkan penuh.

### 04 Function `hitung_total` sebagai kontrak perhitungan

`hitung_total` menerima `p_jumlah` dan `p_harga`, lalu mengembalikan `NUMBER`.

Jumlah harus bukan `NULL`, positif, integer, dan sesuai kapasitas `NUMBER(6)`. Harga harus bukan `NULL`, positif, maksimal dua desimal, dan sesuai kapasitas `NUMBER(12,2)`. Harga 100.123 ditolak sebelum multiplication atau assignment ke kolom, bukan dibulatkan untuk membuat tes terlihat sukses.

### POIN PEMBAHASAN

- Fungsi tidak melakukan DML atau `COMMIT` sehingga bisa dipanggil `SELECT` untuk subtotal.
- Saat proses menggunakan harga katalog, fungsi tetap memvalidasinya dan memastikan satu aturan perhitungan.
- Harga detail disalin saat transaksi terjadi untuk menyimpan histori. `v_total` memperlihatkan pemakaian fungsi dalam procedure walau total tidak disimpan sebagai kolom redundant; view menghitung ulang dari detail historis yang sama.

## 05 Procedure proses dan stock lock

proses menerima pelanggan, produk, jumlah, serta `p_id` OUT.

Validasi awal menolak jumlah pecahan dan ID tidak sah. `SELECT INTO` pelanggan membedakan pelanggan tidak ditemukan melalui `NO_DATA_FOUND`, yang diterjemahkan menjadi error bisnis. Produk diambil dengan `SELECT` harga, stok `FOR UPDATE`; transaksi lain yang meminta lock produk sama harus menunggu sampai pemilik pertama selesai.

### POIN PEMBAHASAN

- Setelah lock diperoleh, stok dibandingkan dengan jumlah.
- Jika cukup, `NEXTVAL` menghasilkan ID, header/detail ditulis dan stok dikurangi.
- Tidak ada `MAX(id)+1`, penjualan anonim baru, atau helper yang mengurangi stok di luar transaksi.
- Lock dapat tetap tertahan setelah error bisnis bila pemanggil belum rollback.
- Error lain seperti duplicate key atau kegagalan trigger diteruskan, bukan dicetak lalu dianggap berhasil.
- Nilai OUT dari pemanggilan gagal tidak boleh dijadikan bukti ID penjualan valid.

## 06 Audit, cursor, dan pemilik transaksi

`UPDATE` stok memicu `trg_audit_stok_demo` yang menulis stok lama/baru ke `audit_stok_demo` tanpa commit.

Perubahan utama dan audit menjadi satu transaksi. `lapor_stok` menggunakan cursor FOR bernama untuk mencetak semua produk terurut; tidak melakukan perubahan data atau kontrol transaksi. Cursor FOR menutup cursor otomatis, tetapi penutupan itu tidak melepaskan transaction lock dari proses.

### POIN PEMBAHASAN

- Package tidak melakukan `COMMIT` atau `ROLLBACK` penuh secara rahasia.
- Contoh pemanggil mandiri memang memiliki seluruh transaksi, sehingga exception menyebabkan rollback seluruh unit kerjanya.
- Uji sukses juga rollback untuk menjaga seed.
- Pilihan penyimpanan permanen ditunjukkan terpisah dan hanya dijalankan setelah pengujian serta keputusan sadar.
- Savepoint pada test harness milik pemanggil membatasi tiap percobaan gagal, bukan kebijakan tersembunyi API.

## 07 Matriks uji dan bukti yang dapat diperiksa

**Jalur sukses memeriksa ID keluaran, detail, stok turun dua, total view, dan satu audit tambahan.**

Setelah rollback, header/detail baru harus hilang, stok kembali, dan jumlah audit pulih. Penjualan 101–103 tetap ada. Harga Kopi pada seed adalah 30000; assertion total 60000 sengaja menguji dataset tersebut. Jika pengguna sudah mengubah data permanen, sesuaikan pengujian dengan nilai awal yang didokumentasikan tanpa memalsukan seed.

### POIN PEMBAHASAN

- Jalur gagal mencakup `NULL`, nol, negatif, pecahan, pelanggan/produk tidak ditemukan, stok kosong, dan nominal tiga desimal.
- Harness membandingkan `SQLCODE` yang diharapkan serta menolak kasus yang tidak mengeluarkan exception.
- Hitungan sebelum/sesudah dibaca pada sesi latihan tanpa penulis lain.
- Nomor sequence boleh berlubang setelah rollback.

- Pengujian dua sesi opsional menunjukkan lock, tetapi jangan menjalankannya bila platform tidak mendukung koneksi paralel.

## 08 Presentasi dan batas verifikasi

**Presentasi menunjukkan ERD seluruh tabel, alur API, daftar objek, query laporan pelanggan dan pegawai, specification/body, hasil `USER_ERRORS`, serta log uji aktual.**

Pisahkan prediksi materi, validasi struktur JSON, dan eksekusi Oracle. Tidak ada klaim account-bound FreeSQL telah diuji sebelum mahasiswa menjalankannya sendiri; kompatibilitas rancangan Oracle 19c tidak sama dengan bukti kompilasi Oracle.

### POIN PEMBAHASAN

- Bahas kekurangan proyek: belum ada multi-item cart, pembayaran, idempotency, otorisasi aplikasi, audit tahan manipulasi, pembatalan stok terkelola, atau strategi deadlock untuk banyak produk.
- `NULL` dan string kosong harus ditafsirkan secara konsisten dalam laporan, bukan disulap menjadi pelanggan terdaftar.
- Artefak akhir berupa script lengkap serta bukti tes dan penjelasan transaction owner, bukan tangkapan layar pesan `DBMS_OUTPUT` saja.

## Alur unit kerja penjualan

- 1 Periksa pelanggan dan produk
- 2 Kunci produk serta validasi jumlah
- 3 Ambil ID sequence
- 4 `INSERT` header/detail dan `UPDATE` stok
- 5 Audit otomatis; caller memilih `COMMIT` / `ROLLBACK`

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.

### Penjualan dua Kopi: pekerjaan berhasil belum tentu tersimpan

Setelah seluruh objek prasyarat siap, pemanggil menjual 2 Kopi dengan harga seed 30000. Contoh dimulai tanpa pekerjaan tertunda.

| Keadaan                                       | Stok Kopi | Detail tambahan | Nilai transaksi tambahan |
|-----------------------------------------------|-----------|-----------------|--------------------------|
| Sebelum proses                                | 20        | 0               | 0                        |
| Setelah package;<br>belum <code>COMMIT</code> | 18        | 1               | 60000                    |
| Jika pemanggil<br><code>ROLLBACK</code>       | 20        | 0               | 0                        |

| Masukan jumlah | Hasil yang diprediksi                 | Alasan               |
|----------------|---------------------------------------|----------------------|
| 0              | ORA-20031                             | Jumlah tidak positif |
| 2              | Berhasil; caller menentukan transaksi | Jumlah memenuhi stok |
| 30             | ORA-20035                             | Stok tidak cukup     |

### Cara membaca contoh

1. Package memvalidasi masukan, mencari pelanggan/produk dan mengunci baris stok.
2. Header/detail, pengurangan stok dan audit termasuk pekerjaan yang sama.
3. Pemanggil memilih **COMMIT** untuk menyimpan atau **ROLLBACK** untuk membatalkan, bukan mengandalkan output pesan.

**Yang perlu dipahami:** Jika jumlah diminta 30 melebihi stok seed 20, package mengembalikan ORA-20035. Tidak ada penjualan berhasil yang boleh dianggap tersimpan. ID berasal dari sequence dan tidak dipastikan selalu 200.

## Video pembelajaran

### Video utama — fokus bab ini

Project Oracle SQL/PLSQL bertahap: tabel relasional, DML, JOIN dan logika transaksi pada sistem perbankan. Menggantikan tutorial satu procedure.

**Oracle SQL & PL/SQL Full Banking Project | Step-by-Step with Real-Time Example**  
Udayit Tech Academy · [Buka video utama di YouTube](#)

### Video tambahan sesuai subtopik

1. [Real-World Oracle PL/SQL for Modern Developers](#)  
Koenig Solutions

**Fokus:** Pendamping penerapan procedure, function, trigger, package dan error handling pada sistem Oracle.

Referensi praktik integrasi; bukan implementasi persis pkg\_penjualan\_demo.

**Batas cakupan dan panduan menonton:** Domain video adalah perbankan, sedangkan tugas kelas tetap penjualan. Gunakan pola integrasi dan pengujian, bukan menyalin schema banking. Kelengkapan seluruh komponen final project dinilai dari implementasi penjualan pada halaman, bukan dari menonton video.

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 — Preflight objek dan constraint proyek

```

SELECT table_name,constraint_name,constraint_type,status FROM
user_constraints
  WHERE table_name IN
('PELANGGAN_DEMO','PRODUK_DEMO','PENJUALAN_DEMO','DETAIL_JUAL_DEMO',
'PEGAWAI_DEMO')
  ORDER BY table_name,constraint_name;
SELECT object_name,object_type,status FROM user_objects
  WHERE object_name IN
('SEQ_PENJUALAN_DEMO','V_TOTAL_PENJUALAN_DEMO','AUDIT_STOK_DEMO','
TRG_AUDIT_STOK_DEMO')
  ORDER BY object_name;
SELECT trigger_name,status FROM user_triggers WHERE
trigger_name='TRG_AUDIT_STOK_DEMO';
SELECT sequence_name,min_value,max_value,increment_by,cycle_flag
FROM user_sequences
  WHERE sequence_name='SEQ_PENJUALAN_DEMO';
SELECT id_jual,id_pelanggan,status FROM penjualan_demo WHERE
id_jual IN (101,102,103) ORDER BY id_jual;

```

Jika dependensi belum tersedia, jalankan definisi penuh modul 13 sekali. Jangan DROP setup. Query adalah bukti keberadaan, bukan administrasi DBA.

## Langkah 2 — Package final lengkap: specification dan body

```

CREATE OR REPLACE PACKAGE pkg_penjualan_demo AS
  FUNCTION hitung_total(p_jumlah IN NUMBER,p_harga IN NUMBER)
RETURN NUMBER;
  PROCEDURE proses(p_pelanggan IN NUMBER,p_produk IN
NUMBER,p_jumlah IN NUMBER,p_id OUT NUMBER);
  PROCEDURE lapor_stok;
END pkg_penjualan_demo;
/
CREATE OR REPLACE PACKAGE BODY pkg_penjualan_demo AS
  FUNCTION hitung_total(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(-20031,'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(-20032,'Harga harus positif, maksimal
dua desimal dan sesuai kapasitas kolom');
END IF;
RETURN p_jumlah*p_harga;
END hitung_total;
PROCEDURE proses(p_pelanggan IN NUMBER,p_produk IN
NUMBER,p_jumlah IN NUMBER,p_id OUT NUMBER) IS
    v_pelanggan pelanggan_demo.id_pelanggan%TYPE;
    v_harga produk_demo.harga%TYPE;
    v_stok produk_demo.stok%TYPE;
    v_total NUMBER;
BEGIN
    p_id := NULL;
    IF p_jumlah IS NULL OR p_jumlah <= 0 OR p_jumlah <>
TRUNC(p_jumlah) OR p_jumlah > 999999 THEN
        RAISE_APPLICATION_ERROR(-20031,'Jumlah harus integer positif
maksimal 999999');
    END IF;
    IF p_pelanggan IS NULL OR p_pelanggan <= 0 OR p_pelanggan <>
TRUNC(p_pelanggan) THEN
        RAISE_APPLICATION_ERROR(-20033,'Pelanggan tidak valid');
    END IF;
    IF p_produk IS NULL OR p_produk <= 0 OR p_produk <>
TRUNC(p_produk) THEN
        RAISE_APPLICATION_ERROR(-20034,'Produk tidak valid');
    END IF;
    BEGIN
        SELECT id_pelanggan INTO v_pelanggan FROM pelanggan_demo WHERE
id_pelanggan=p_pelanggan;
    EXCEPTION WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20033,'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(-20034,'Produk tidak ditemukan');
END;
v_total := hitung_total(p_jumlah,v_harga);
IF v_stok < p_jumlah THEN RAISE_APPLICATION_ERROR(-20035,'Stok
tidak cukup'); END IF;
SELECT seq_penjualan_demo.NEXTVAL INTO p_id FROM dual;
INSERT INTO penjualan_demo(id_jual,id_pelanggan,tanggal,status)
VALUES(p_id,v_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 proses;
PROCEDURE lapor_stok IS
    CURSOR c_stok IS SELECT id_produk,nama_produk,harga,stok FROM
produk_demo ORDER BY id_produk;
BEGIN
    FOR r IN c_stok LOOP
        DBMS_OUTPUT.PUT_LINE(r.id_produk || ': ' || r.nama_produk || ',
harga=' || r.harga || ', stok=' || r.stok);
    END LOOP;
END lapor_stok;
END pkg_penjualan_demo;
/

```

Function, procedure proses, cursor report, validasi, lock, INSERT dan UPDATE seluruhnya terlihat. Tidak ada commit/rollback tersembunyi di package.

### Langkah 3 — Periksa kompilasi sebelum melakukan transaksi

```

SELECT name, type, line, position, text
FROM user_errors
WHERE name IN ('PKG_PENJUALAN_DEMO','TRG_AUDIT_STOK_DEMO')
ORDER BY name, type, sequence;
SELECT object_name,object_type,status FROM user_objects WHERE
object_name IN
('PKG_PENJUALAN_DEMO','TRG_AUDIT_STOK_DEMO','V_TOTAL_PENJUALAN_DEM
O') ORDER BY object_name,object_type;

```

Lanjutkan hanya setelah PACKAGE dan PACKAGE BODY hadir VALID, trigger VALID/enabled, dan USER\_ERRORS tidak menunjukkan error. Kompilasi Oracle belum diverifikasi oleh penulis materi.

## Langkah 4 — Pemanggil sukses, assertion, dan rollback seed

```
-- Pemanggil memiliki seluruh transaksi; mulai tanpa pekerjaan
-- tertunda.
DECLARE
  v_id NUMBER;
  v_stok_awal NUMBER;
  v_stok_akhir NUMBER;
  v_audit_awal NUMBER;
  v_audit_akhir NUMBER;
  v_total NUMBER;
  v_jumlah NUMBER;
BEGIN
  SELECT stok INTO v_stok_awal FROM produk_demo WHERE id_produk=11;
  SELECT COUNT(*) INTO v_audit_awal FROM audit_stok_demo WHERE
id_produk=11;
  pkg_penjualan_demo.proses(1,11,2,v_id);
  SELECT stok INTO v_stok_akhir FROM produk_demo WHERE
id_produk=11;
  SELECT total INTO v_total FROM v_total_penjualan_demo WHERE
id_jual=v_id;
  SELECT COUNT(*) INTO v_audit_akhir FROM audit_stok_demo WHERE
id_produk=11;
  IF v_stok_akhir <> v_stok_awal-2 OR v_audit_akhir <>
v_audit_awal+1 OR v_total <> 60000 THEN
    RAISE_APPLICATION_ERROR(-20991,'Uji sukses tidak memenuhi
stok/audit/total seed');
  END IF;
  DBMS_OUTPUT.PUT_LINE('ID=' || v_id || ', total=' || v_total);
  pkg_penjualan_demo.lapor_stok;
  -- Kebijakan uji: batalkan jalur sukses agar seed utuh.
  ROLLBACK;
  SELECT COUNT(*) INTO v_jumlah FROM detail_jual_demo WHERE
id_jual=v_id;
  IF v_jumlah <> 0 THEN RAISE_APPLICATION_ERROR(-20992,'Detail uji
masih ada setelah rollback'); END IF;
  SELECT COUNT(*) INTO v_jumlah FROM penjualan_demo WHERE
```

```

id_jual=v_id;
  SELECT stok INTO v_stok_akhir FROM produk_demo WHERE
id_produk=11;
  SELECT COUNT(*) INTO v_audit_akhir FROM audit_stok_demo WHERE
id_produk=11;
  IF v_jumlah <> 0 OR v_stok_akhir <> v_stok_awal OR v_audit_akhir
<> v_audit_awal THEN
    RAISE_APPLICATION_ERROR(-20992,'Rollback sukses tidak memulihkan
data/audit');
  END IF;
  DBMS_OUTPUT.PUT_LINE('Rollback sukses: header hilang, stok dan
audit kembali');
EXCEPTION WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END;
/

```

Pada seed total 60000, stok Kopi sementara 18 lalu kembali 20. ID tidak harus 200. Rollback juga menghapus detail melalui pembatalan INSERT transaksi, bukan melalui DELETE cascade.

## Langkah 5 — Pemanggil gagal lengkap dengan kode error dan rollback

```

-- Setiap kasus memakai savepoint milik pemanggil dan memeriksa
kode error.
DECLARE
  v_id NUMBER;
  v_total NUMBER;
  v_header_awal NUMBER;
  v_detail_awal NUMBER;
  v_audit_awal NUMBER;
  v_n NUMBER;
  PROCEDURE uji_gagal(p_pelanggan NUMBER,p_produk NUMBER,p_jumlah
NUMBER,p_error NUMBER) IS
  BEGIN
    SAVEPOINT kasus_14;
    BEGIN
      pkg_penjualan_demo.proses(p_pelanggan,p_produk,p_jumlah,v_id);

```

```

EXCEPTION WHEN OTHERS THEN
    IF SQLCODE <> p_error THEN RAISE; END IF;
    DBMS_OUTPUT.PUT_LINE('Ditolak sesuai harapan: ' || SQLERRM);
    ROLLBACK TO kasus_14;
    RETURN;
END;
ROLLBACK TO kasus_14;
RAISE_APPLICATION_ERROR(-20993,'Kasus invalid justru diterima');
END;
PROCEDURE uji_harga(p_harga NUMBER) IS
BEGIN
    BEGIN
        v_total := pkg_penjualan_demo.hitung_total(1,p_harga);
    EXCEPTION WHEN OTHERS THEN
        IF SQLCODE <> -20032 THEN RAISE; END IF;
        DBMS_OUTPUT.PUT_LINE('Harga ditolak: ' || SQLERRM);
        RETURN;
    END;
    RAISE_APPLICATION_ERROR(-20994,'Harga invalid justru diterima');
END;
BEGIN
    SELECT COUNT(*) INTO v_header_awal FROM penjualan_demo;
    SELECT COUNT(*) INTO v_detail_awal FROM detail_jual_demo;
    SELECT COUNT(*) INTO v_audit_awal FROM audit_stok_demo;
    uji_gagal(1,11,NULL,-20031);
    uji_gagal(1,11,0,-20031);
    uji_gagal(1,11,-1,-20031);
    uji_gagal(1,11,1.5,-20031);
    uji_gagal(NULL,11,1,-20033);
    uji_gagal(9999,11,1,-20033);
    uji_gagal(1,9999,1,-20034);
    uji_gagal(1,14,1,-20035);
    uji_harga(NULL);
    uji_harga(0);
    uji_harga(-100);
    uji_harga(100.123);
    ROLLBACK;
    SELECT COUNT(*) INTO v_n FROM penjualan_demo;
    IF v_n <> v_header_awal THEN
        RAISE_APPLICATION_ERROR(-20995,'Header berubah'); END IF;
    SELECT COUNT(*) INTO v_n FROM detail_jual_demo;

```

```

IF v_n <> v_detail_awal THEN
RAISE_APPLICATION_ERROR(-20995,'Detail berubah'); END IF;
SELECT COUNT(*) INTO v_n FROM audit_stok_demo;
IF v_n <> v_audit_awal THEN RAISE_APPLICATION_ERROR(-20995,'Audit
berubah'); END IF;
DBMS_OUTPUT.PUT_LINE('Semua kasus gagal sesuai harapan; jumlah
data/audit tetap');
EXCEPTION WHEN OTHERS THEN
ROLLBACK;
RAISE;
END;
/

```

Harness tidak menelan error sembarang. Helper lokal merupakan bagian pemanggil pengujian, bukan anggota package bisnis. Uji produk 14 mengandalkan stok seed nol.

## Langkah 6 — Laporan menggunakan seluruh tabel, view, dan function

```

SELECT c.id_pelanggan,c.nama,COUNT(p.id_jual) AS jumlah_penjualan,
      NVL(SUM(v.total),0) AS total_penjualan
FROM pelanggan_demo c LEFT JOIN penjualan_demo p ON
p.id_pelanggan=c.id_pelanggan
LEFT JOIN v_total_penjualan_demo v ON v.id_jual=p.id_jual
GROUP BY c.id_pelanggan,c.nama ORDER BY c.id_pelanggan;
SELECT p.id_jual,NVL(c.nama,'Anonim') AS
pelanggan,d.id_produk,pr.nama_produk,
      pkg_penjualan_demo.hitung_total(d.jumlah,d.harga) AS
subtotal
FROM penjualan_demo p LEFT JOIN pelanggan_demo c ON
c.id_pelanggan=p.id_pelanggan
JOIN detail_jual_demo d ON d.id_jual=p.id_jual
JOIN produk_demo pr ON pr.id_produk=d.id_produk ORDER BY
p.id_jual,d.id_produk;
SELECT LEVEL AS tingkat,id_pegawai,nama,id_manajer
FROM pegawai_demo START WITH id_manajer IS NULL
CONNECT BY PRIOR id_pegawai=id_manajer ORDER SIBLINGS BY
id_pegawai;
BEGIN
  pkg_penjualan_demo.lapor_stok;

```

```
END;
/
```

Laporan pelanggan mempertahankan Citra; laporan detail mempertahankan sale anonim 103. Hierarki pegawai memberi konteks pengelola tanpa menciptakan relasi penjualan yang tidak ada.

## Langkah 7 — Pilihan eksplisit: pemanggil menyimpan transaksi nyata

```
-- OPSIONAL: bukan bagian uji rollback. Menambah penjualan dan
audit permanen.
-- Jalankan hanya di schema latihan sendiri setelah menyetujui
perubahan seed.
DECLARE
  v_id NUMBER;
BEGIN
  pkg_penjualan_demo.proses(1,11,1,v_id);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Penjualan disimpan dengan ID=' || v_id);
EXCEPTION WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END;
/
SELECT * FROM v_total_penjualan_demo ORDER BY id_jual;
SELECT id_produk,stok FROM produk_demo WHERE id_produk=11;
SELECT id_produk,operasi,stok_lama,stok_baru,waktu FROM
audit_stok_demo ORDER BY waktu;
```

Pemanggil, bukan package, melakukan COMMIT. Setelah commit perubahan ini tidak dapat dibatalkan dengan ROLLBACK; lewati cell ini bila ingin seed tetap persis awal.

## Tugas / latihan

1. Kerjakan pada XE lokal ATAU FreeSQL login schema sendiri; tampilkan ERD dan constraint seluruh lima tabel, tanpa tugas administrasi wajib.

2. Gunakan dependensi modul 13 dan bangun pkg\_penjualan\_demo dengan signature tepat; sertakan seluruh specification/body, no hidden commit.
3. Jalankan preflight, USER\_ERRORS, pemanggil sukses/gagal, laporan, dan assertion rollback; lampirkan hasil aktual serta ID sequence yang mungkin berlubang.
4. Presentasikan fungsi, procedure, cursor, exception, trigger, sequence/view, dan ownership transaksi, termasuk batas sistem bukan production-safe.
5. Kumpulkan script lengkap, matriks expected/actual, bukti seed 101–103 tetap ada, dan penjelasan apakah cell COMMIT opsional dilewati.

## Pertanyaan analisis

1. Jika pemanggil menangkap exception proses lalu meneruskan transaksi tanpa rollback, risiko apa yang terjadi pada lock dan DML yang sudah berhasil?
2. Mengapa constraint, validasi PL/SQL, audit, dan assertion pemanggil saling melengkapi tetapi tidak membuat sistem siap produksi?

## 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.

**Presentasi project:** jelaskan skema, jalankan query/proses, tampilkan kasus sukses dan gagal, lalu terangkan integritas serta transaksi.

## 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 Packages](#)
- [Oracle 19c — PL/SQL Subprograms](#)
- [Oracle 19c — Cursors Overview](#)
- [Oracle 19c — PL/SQL Triggers](#)
- [Oracle 19c — CREATE SEQUENCE](#)
- [Oracle 19c — CREATE VIEW](#)
- [Oracle 19c — SELECT, FOR UPDATE](#)
- [Oracle 19c — COMMIT](#)
- [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.

---