Jobsheet Praktikum Kelompok: Pertemuan 4

Desain Basis Data, Migrasi, dan Seeding (Kerja Kelompok)

Mata Kuliah	Pemrograman Web Lanjut (SIB245007)
Pertemuan	4 (Minggu 4)
Durasi	2 sesi × 170 menit
Sub-CPMK	Sub-CPMK 1: Mahasiswa mampu memahami konsep dasar web framework serta menerapkan routing, controller, dan pengelolaan basis data dalam pengembangan aplikasi web.
Mode Pengerjaan	Kelompok (sama seperti Pertemuan 3), satu repositori GitHub bersama per kelompok
Kode Awal	repositori kelompok hasil Pertemuan 3 (lanjutkan `main`-nya)

A. Capaian Praktikum

Setelah menyelesaikan jobsheet kelompok ini, kamu mampu:

1. Merancang skema empat tabel inti Simple POS (`categories`, `products`, `transactions`, `transaction_details`) beserta relasi foreign key-nya.
2. Menulis migrasi untuk skema itu, mengikuti aturan emas "migrasi baru, jangan edit migrasi lama yang sudah pernah dijalankan".
3. Menulis seeder berskala nyata (300 produk, 2.500 transaksi) memakai bulk insert `DB::table()->insert()` dalam batch, bukan `Model::create()` satu per satu, dengan insert transaksi dan detailnya dibungkus `DB::transaction()`.
4. Membuktikan dampak index pada kolom foreign key lewat `EXPLAIN QUERY PLAN`, dari `SCAN` (memindai seluruh tabel) menjadi `SEARCH ... USING INDEX`.
5. Mengganti data produk hardcoded di `TransactionController` (dari Pertemuan 3) dengan Model `Product` yang membaca dari basis data hasil seeding.

B. Persiapan dan Prasyarat

- **Alat**: sama seperti Pertemuan 3 (PHP 8.2+, Composer, Node.js, Git), ditambah `sqlite3` CLI kalau ada (opsional, ada jalur alternatif lewat Tinker kalau tidak terpasang).
- **Identitas git**: sudah diatur sejak Pertemuan 3. Kalau ganti laptop atau komputer lab, ulangi pengecekan `git config --global user.name`/`user.email` seperti pada jobsheet itu.
- **Kelanjutan kode**: kelompok melanjutkan repositori GitHub bersama dari Pertemuan 3, dari `main` yang sudah berisi `increment 3`. Kalau repositori kelompok bermasalah atau tantangan mandiri Pertemuan 3 belum selesai, mulai dari template `simple-pos-ch03` (`https://github.com/se-polinema/simple-pos-ch03`, dibuat lewat **Use this template** seperti Langkah 1 Pertemuan 3), lalu ulangi Langkah 5 Pertemuan 3 (menyederhanakan `TransactionController`) sebelum melanjutkan jobsheet ini, karena `simple-pos-ch03` berisi versi produksi controller yang memanggil Model yang belum kamu buat.
- **Verifikasi cepat** sebelum mulai:

```
git checkout main
git pull
git log --oneline -1
```

> ✓ **Checkpoint:** baris teratas menunjukkan commit `increment 3: ...`.

C. Langkah Kerja

Alur kerja sama seperti Pertemuan 3: setiap langkah pembangunan (Langkah 1 sampai 5) dikerjakan di branch terpisah dari `main`, digabungkan lewat Pull Request dengan **Create a merge**

commit (bukan squash), lalu semua anggota `git pull` sebelum membuat branch berikutnya. Rancang dulu skemanya di atas kertas sebelum menulis migrasi apa pun.

Langkah 1: Merancang skema di atas kertas

Sebelum menulis kode, kelompok menggambar skema berikut di atas kertas atau papan tulis dan menyepakati siapa mengerjakan migrasi tabel yang mana:

Tabel	Kolom Utama
`categories`	`id`, `name`
`products`	`id`, `category_id` (fk), `name`, `price`, `stock`
`transactions`	`id`, `user_id` (fk), `total`, `created_at`
`transaction_details`	`id`, `transaction_id` (fk), `product_id` (fk), `qty`, `subtotal`

Satu `categories` punya banyak `products`. Satu `products` bisa muncul di banyak `transaction_details`, dan satu `transactions` punya banyak `transaction_details`, masing-masing mencatat satu baris item yang dibeli. `transactions` menunjuk balik ke `users` (kasir yang memproses transaksi itu, sudah ada sejak Pertemuan 1).

Perhatikan `transactions.total` dan `transaction_details.subtotal`: keduanya kolom yang **disimpan**, bukan dihitung ulang setiap kali dibaca. Itu keputusan sadar, karena total transaksi harus tetap sama persis seperti nilai saat transaksi itu terjadi, meskipun harga produk berubah di kemudian hari. Pertemuan validasi nanti membahas kenapa nilai ini juga wajib dihitung ulang di server saat disimpan, bukan sekadar dipercaya dari input.

> ✓ **Checkpoint:** kelompok sudah punya gambar skema di atas kertas/papan, dan pembagian tugas Langkah 2-5 sudah disepakati.

Langkah 2: Migrasi `categories` dan `products`

Buat branch baru dari `main` terbaru, misalnya `categories-products-migration`:

```
git checkout main
git pull
git checkout -b categories-products-migration
php artisan make:model Category -m
php artisan make:migration create_products_table
```

`make:model Category -m` membuat model `Category` sekaligus berkas migrasinya dalam satu perintah; model ini dipakai nanti oleh seeder di Langkah 4. Model `Product` sengaja belum dibuat sekarang, itu bagian dari Langkah 5.

Isi metode `up()` migrasi `categories`:

```
Schema::create('categories', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->timestamps();
});
```

Isi metode `up()` migrasi `products`:

```
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->foreignId('category_id')->constrained();
    $table->string('name');
    $table->unsignedInteger('price');
    $table->unsignedInteger('stock')->default(0);
    $table->timestamps();
});
```

`foreignId('category_id')->constrained()` adalah pasangan yang sering dipakai bersama: baris pertama membuat kolom `category_id` bertipe unsigned big integer, baris kedua menambahkan *constraint* foreign key yang menunjuk ke `categories.id` berdasarkan konvensi penamaan

Laravel. Constraint ini menjaga integritas data, mencegah baris produk menunjuk ke kategori yang sudah dihapus, tapi seperti akan terlihat di Langkah 4, itu tidak otomatis berarti kolomnya sudah punya index.

```
git add .
git commit -m "tambah migrasi categories dan products"
git push -u origin categories-products-migration
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✅ **Checkpoint:** `php artisan migrate:fresh` (tanpa `--seed` dulu) berjalan tanpa error, dan `php artisan tinker` lalu `Schema::hasTable('products')` mengembalikan `true`.

Langkah 3: Migrasi `transactions` dan `transaction_details`

Buat branch baru dari `main` terbaru, misalnya `transactions-migration`:

```
git checkout main
git pull
git checkout -b transactions-migration
php artisan make:migration create_transactions_table
php artisan make:migration create_transaction_details_table
```

Isi metode `up()` migrasi `transactions`:

```
Schema::create('transactions', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->unsignedInteger('total');
    $table->timestamps();
});
```

Isi metode `up()` migrasi `transaction_details`:

```
Schema::create('transaction_details', function (Blueprint $table) {
    $table->id();
    $table->foreignId('transaction_id')->constrained();
    $table->foreignId('product_id')->constrained();
    $table->unsignedInteger('qty');
    $table->unsignedInteger('subtotal');
    $table->timestamps();
});
```

```
git add .
git commit -m "tambah migrasi transactions dan transaction_details"
git push -u origin transactions-migration
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✅ **Checkpoint:** `php artisan migrate:fresh` berjalan tanpa error dan menampilkan keempat tabel baru (`categories`, `products`, `transactions`, `transaction_details`) di daftar migrasi yang dijalankan.

> ⚠ **Jika gagal:** error `no such table: categories` saat migrasi `products` dijalankan berarti urutan nama berkas migrasi salah, migrasi `categories` harus punya stempel waktu lebih awal dari `products` (dan `transactions` sebelum `transaction_details`), karena Laravel menjalankannya berurutan sesuai nama berkas.

Langkah 4: Seeder skala nyata dan pembuktian index

Buat branch baru, misalnya `seeder-and-index`:

```
git checkout main
git pull
git checkout -b seeder-and-index
```

Ganti isi `database/seeders/DatabaseSeeder.php` menjadi:

```
<?php

namespace Database\Seeders;
```

```php
use App\Models\Category;
use App\Models\User;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $user = User::factory()->create([
            'name' => 'Test User',
            'email' => 'test@example.com',
        ]);

        $categoryIds = collect(['Makanan', 'Minuman', 'Snack', 'Lainnya'])
            ->map(fn (string $name) => Category::create(['name' => $name])->id)
            ->all();

        $products = [];
        foreach ($categoryIds as $categoryId) {
            for ($i = 0; $i < 75; $i++) {
                $products[] = [
                    'category_id' => $categoryId,
                    'name' => fake()->words(2, true),
                    'price' => fake()->numberBetween(3000, 50000),
                    'stock' => fake()->numberBetween(0, 200),
                    'created_at' => now(),
                    'updated_at' => now(),
                ];
            }
        }

        foreach (array_chunk($products, 50) as $chunk) {
            DB::table('products')->insert($chunk);
        }

        $productPrices = DB::table('products')->pluck('price', 'id');
        $productIds = $productPrices->keys()->all();

        for ($t = 0; $t < 2500; $t++) {
            DB::transaction(function () use ($user, $productIds, $productPrices) {
                $itemCount = fake()->numberBetween(1, 4);
                $total = 0;
                $details = [];

                for ($i = 0; $i < $itemCount; $i++) {
                    $productId = fake()->randomElement($productIds);
                    $qty = fake()->numberBetween(1, 3);
                    $subtotal = $productPrices[$productId] * $qty;
                    $total += $subtotal;

                    $details[] = [
                        'product_id' => $productId,
                        'qty' => $qty,
                        'subtotal' => $subtotal,
                    ];
                }

                $transactionId = DB::table('transactions')->insertGetId([
                    'user_id' => $user->id,
                    'total' => $total,
                    'created_at' => now(),
                    'updated_at' => now(),
                ]);
```

```
                foreach ($details as &$detail) {
                    $detail['transaction_id'] = $transactionId;
                    $detail['created_at'] = now();
                    $detail['updated_at'] = now();
                }

                DB::table('transaction_details')->insert($details);
            });
        }
    }
}
```

Baris `User::factory()->create(['name' => 'Test User', ...])` di atas persis dengan pengguna yang sudah dibuat seeder sejak Pertemuan 1, hanya dipindah ke sini supaya seluruh isi berkas terlihat lengkap. `DB::table()->insert()` melewati lapisan Eloquent sama sekali: ia menyusun satu pernyataan `INSERT` yang menulis banyak baris sekaligus, jauh lebih sedikit round-trip ke basis data dibanding memanggil `Model::create()` satu per satu di dalam loop 300 kali. `array_chunk()` membagi array besar menjadi kelompok-kelompok kecil sebelum di-insert, karena sebagian mesin basis data punya batas jumlah baris atau parameter dalam satu pernyataan `INSERT`. Harga produk diambil sekali di awal lewat `pluck('price', 'id')` menjadi array asosiatif, bukan di-query ulang untuk setiap item transaksi, konsisten dengan alasan bulk insert dipakai: makin sedikit round-trip ke basis data, makin cepat seeder berjalan. `DB::transaction()` membungkus insert `transactions` dan `transaction_details`-nya: kalau prosesnya gagal di tengah jalan, seluruh perubahan dalam blok itu dibatalkan bersama, sehingga basis data tidak pernah berakhir dengan transaksi yang tercatat tapi detailnya hilang separuh.

```
php artisan migrate:fresh --seed
```

> ✔ **Checkpoint:** perintah berjalan tanpa error. Verifikasi jumlah baris lewat `php artisan tinker`:
>
> ```
> >>> DB::table('products')->count();
> => 300
> >>> DB::table('transactions')->count();
> => 2500
> ```

Sekarang buktikan dampak index. Lihat kondisi **sebelum** ada index eksplisit:

```
sqlite3 database/database.sqlite \
  "EXPLAIN QUERY PLAN SELECT * FROM products WHERE category_id = 3;"
```

> ✔ **Checkpoint:** baris hasilnya memuat kata `SCAN products`, tandanya SQLite memindai seluruh tabel dari baris pertama sampai terakhir untuk menemukan baris yang cocok.

> ⚠ **Jika gagal:** perintah `sqlite3` tidak dikenali berarti CLI-nya belum terinstal di sistem. Pakai jalur alternatif lewat Tinker tanpa perlu memasang apa pun:
>
> ```
> php artisan tinker
> >>> DB::select("EXPLAIN QUERY PLAN SELECT * FROM products WHERE category_id = 3");
> ```
>
> Hasilnya berupa array asosiatif; kolom `detail` di dalamnya berisi teks SCAN/SEARCH yang sama seperti pada CLI `sqlite3`.

Buat migrasi baru khusus untuk index (jangan mengedit migrasi lama yang sudah pernah dijalankan):

```
php artisan make:migration add_index_to_products_and_transactions_table
```

Buka berkas migrasi yang baru dibuat, lalu isi metode `up()`-nya:

```
public function up(): void
{
    Schema::table('products', function (Blueprint $table) {
        $table->index('category_id');
    });

    Schema::table('transactions', function (Blueprint $table) {
        $table->index('user_id');
    });
}
```

Jalankan migrasi itu. Pakai `php artisan migrate`, bukan `php artisan migrate:fresh`, supaya index ditambahkan di atas data seeder yang sudah ada, bukan menghapus semuanya lagi:

```
php artisan migrate
```

Jalankan ulang perintah `EXPLAIN QUERY PLAN` yang persis sama seperti sebelumnya.

> ✓ **Checkpoint:** baris hasilnya sekarang berubah menjadi `SEARCH products USING INDEX products_category_id_index`, tandanya SQLite langsung melompat ke baris yang relevan lewat struktur index, tanpa menyentuh baris lain sama sekali.

```
git add .
git commit -m "tambah seeder skala nyata dan index foreign key"
git push -u origin seeder-and-index
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main`, pull, dan jalankan `php artisan migrate:fresh --seed` di laptop masing-masing supaya datanya sama.

Langkah 5: Mengganti data hardcoded dengan Model `Product`

Jobsheet Pertemuan 3 menjanjikan ini: data produk yang ditulis langsung di `TransactionController` akan diganti dengan Model `Product` sungguhan begitu tabelnya siap. Sekarang saatnya menepati janji itu.

Buat branch baru, misalnya `product-model`:

```
git checkout main
git pull
git checkout -b product-model
php artisan make:model Product
```

Ubah method `create()` di `app/Http/Controllers/TransactionController.php`, dari mengembalikan `collect([...])` hardcoded menjadi:

```
use App\Models\Product;

// ...

public function create()
{
    $products = Product::take(12)->get();

    return view('pos.create', ['products' => $products]);
}
```

Hapus array `collect([...])` hardcoded beserta `use` yang sudah tidak dipakai. Method `store()`, `index()`, dan `show()` tetap seperti Pertemuan 3 untuk sekarang, listing lengkap dengan pagination dibahas pada pertemuan ORM & relasi berikutnya, jadi `take(12)` di sini sekadar membatasi tampilan sementara.

```
git add .
git commit -m "ganti data hardcoded dengan model product"
git push -u origin product-model
```

Buka Pull Request ke `main`, merge, lalu semua anggota kembali ke `main` dan pull.

> ✓ **Checkpoint:** membuka `http://127.0.0.1:8000/pos` menampilkan 12 produk hasil seeding (nama dan harga acak dari `fake()`), bukan lagi enam produk hardcoded dari Pertemuan 3.

> ⚠ **Jika gagal:** error `Class "App\Models\Product" not found` berarti `use App\Models\Product;` belum ditambahkan di bagian atas controller, atau `php artisan make:model Product` belum dijalankan/di-commit.

Langkah 6: Bersama, uji integrasi dan review kode

Semua anggota `git checkout main && git pull`, jalankan `php artisan migrate:fresh --seed`, lalu ulangi uji `EXPLAIN QUERY PLAN` dan buka `/pos` di laptop masing-masing. Setelah itu, kelompok membaca kode bersama: setiap anggota menjelaskan satu migrasi atau bagian seeder yang **bukan** ia tulis sendiri.

> ✓ **Checkpoint:** hasil `EXPLAIN QUERY PLAN` dan tampilan `/pos` sama persis di setiap laptop anggota.

Langkah 7: Tantangan mandiri kelompok dan commit `increment 4`

Bagi tugas berikut di antara anggota, supaya setiap anggota tercatat minimal satu commit bermakna lewat Pull Request masing-masing:

- Tambahkan index pada kolom `transaction_details.product_id` lewat migrasi baru, lalu buktikan dengan `EXPLAIN QUERY PLAN` pada query yang mencari seluruh detail transaksi untuk satu produk tertentu.
- Tambahkan kolom baru `sku` (string, unik, nullable) pada tabel `products` lewat migrasi baru, bukan dengan mengubah migrasi lama yang sudah pernah dijalankan.
- Tambahkan kolom `is_active` (boolean, default `true`) pada tabel `products` lewat migrasi baru, lalu ubah seeder supaya sekitar 10% produk dibuat dengan `is_active` bernilai `false`.
- Tulis satu query listing baru (misalnya transaksi dalam rentang tanggal tertentu) lewat `php artisan tinker`, lalu jalankan `EXPLAIN QUERY PLAN` pada query itu untuk memeriksa apakah ia sudah memakai index yang ada.
- Ubah jumlah produk yang ditampilkan `TransactionController::create()` dari `take(12)` menjadi menampilkan seluruh produk yang `stock`-nya di atas 0.

Setiap tugas: buat branch baru (misalnya `index-transaction-details`), kerjakan, commit, push, lalu buka dan merge Pull Request-nya (ikuti alur Langkah 3 Pertemuan 3).

Setelah semua Pull Request masuk dan digabungkan, salah satu anggota menutup pekerjaan kelompok:

```
git checkout main
git pull
git commit --allow-empty -m "increment 4: skema, seeder, dan index simple pos"
git push
git log --pretty="%h %an %s"
```

> ✓ **Checkpoint:** baris teratas `git log --pretty="%h %an %s"` menunjukkan `increment 4: ...`, dan baris-baris di bawahnya menunjukkan nama setiap anggota kelompok.

D. Tugas dan Deliverable

Kumpulkan hal berikut sesuai format yang diminta dosen:

- Link repositori GitHub kelompok.
- Screenshot output `EXPLAIN QUERY PLAN` sebelum index (`SCAN`) dan sesudah index (`SEARCH ... USING INDEX`).
- Screenshot halaman `/pos` menampilkan produk hasil seeding (bukan lagi data hardcoded).
- Output `git log --pretty="%h %an %s"` yang menunjukkan minimal satu commit per anggota.
- Tabel pembagian tugas: nama anggota | langkah/tugas yang dikerjakan | hash commit.

E. Kriteria Penilaian

Komponen	Bobot	Kriteria Lengkap (100%)	Kriteria Minimum
Langkah kerja tuntas (kelompok)	40%	Langkah 1-7 selesai, seeder berjalan dengan jumlah baris benar, `/pos` menampilkan data dari database	Sebagian besar langkah selesai, seeder dan `/pos` berfungsi

Komponen	Bobot	Kriteria Lengkap (100%)	Kriteria Minimum
Checkpoint terverifikasi (kelompok)	25%	Screenshot `EXPLAIN QUERY PLAN` sebelum/sesudah, tabel pembagian tugas, dan git log lengkap dan benar	Sebagian checkpoint terbukti
Kontribusi per anggota (individu)	25%	Minimal satu commit bermakna atas nama tiap anggota, sesuai tabel pembagian tugas	Commit ada tapi kecil atau kurang jelas kaitannya
Kerapian repositori dan commit	10%	Pesan `increment 4` persis, migrasi baru (bukan edit migrasi lama), tanpa menyertakan `vendor//node_modules//.env`, PR di-merge rapi (bukan squash)	Commit ada, pesan kurang rapi atau PR di-squash